Introduction to Artificial Intelligence

丁尧相
浙江大学

Fall & Winter 2022
Week 3

Announcements

- Course website is out:

  https://yaoxiangding.github.io/introAI-2022/

- We have no lecture next week due to the National Day!

- We will release Problem Set 1.2 after this lecture.

- We will release the first lab project next week :-P

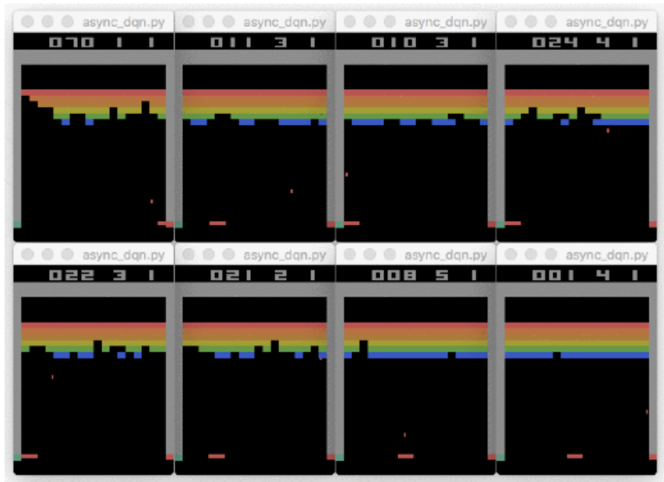
Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-home messages

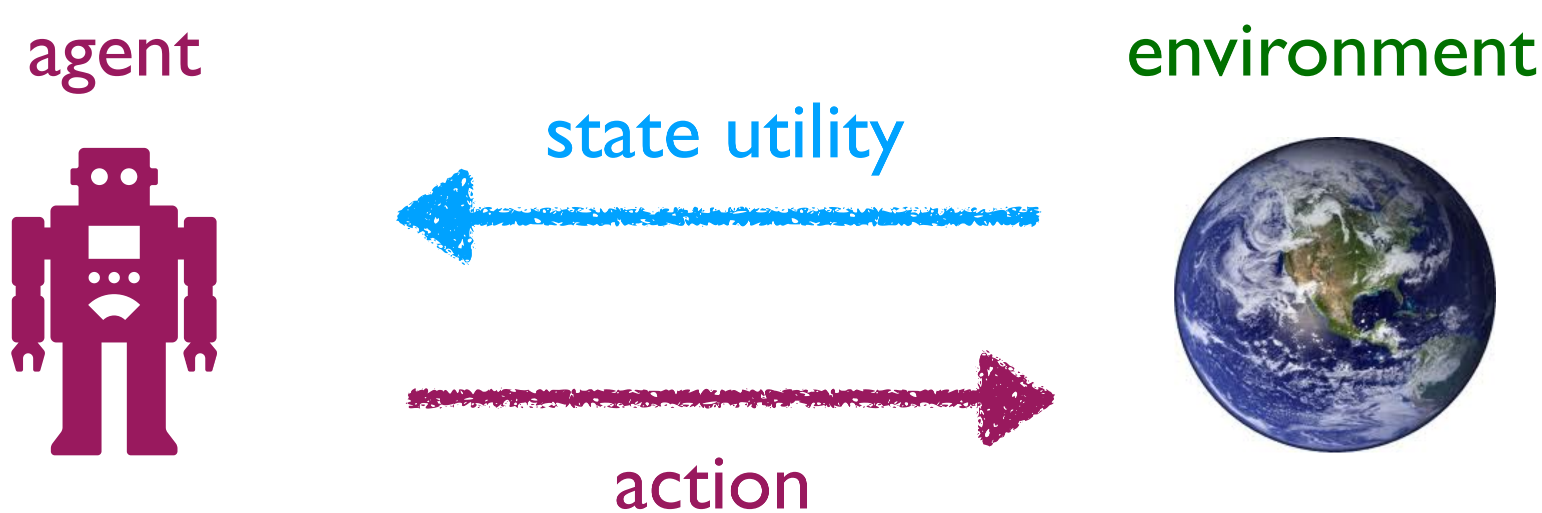
Outline: Decision Making (II)

- Adversarial game
    - Two-player zero-sum game
- Deterministic search
    - Minimax search
    - Alpha-beta pruning
- Simulation-based search
    - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Decision Making

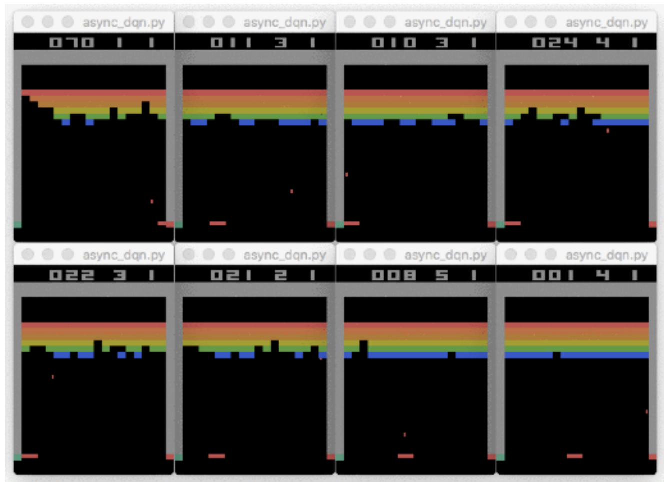


- Conduct action in any state of an environment.

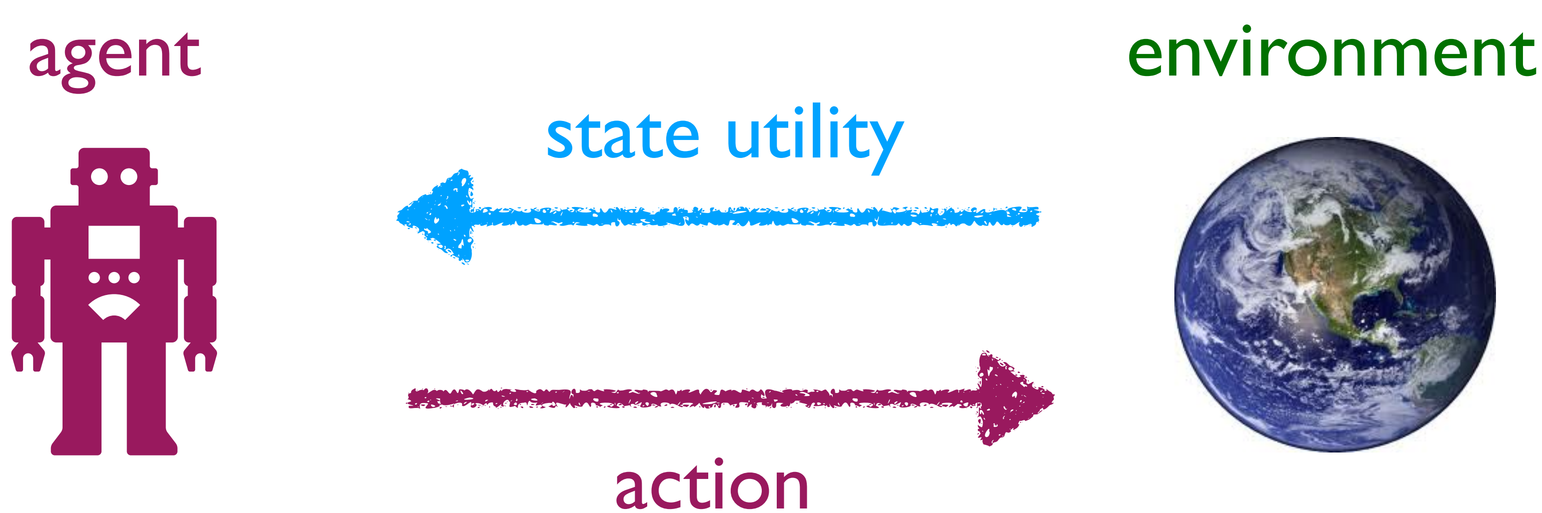


Rational agents make decisions to maximize their own utilities: Games.

Decision Making

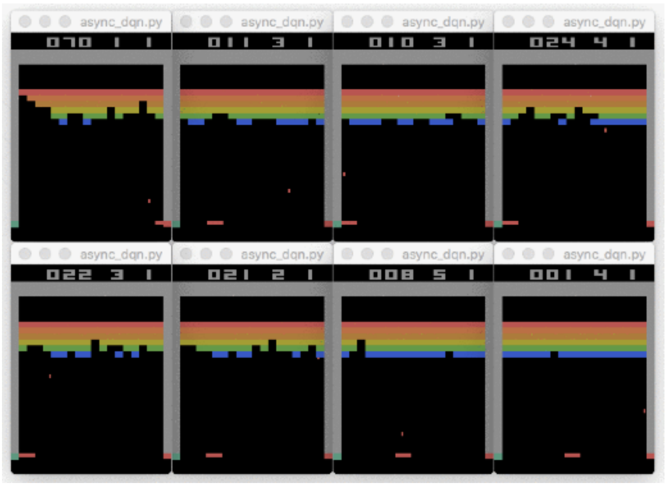


- Conduct action in any state of an environment.



Rational agents make decisions to maximize their own utilities: Games.

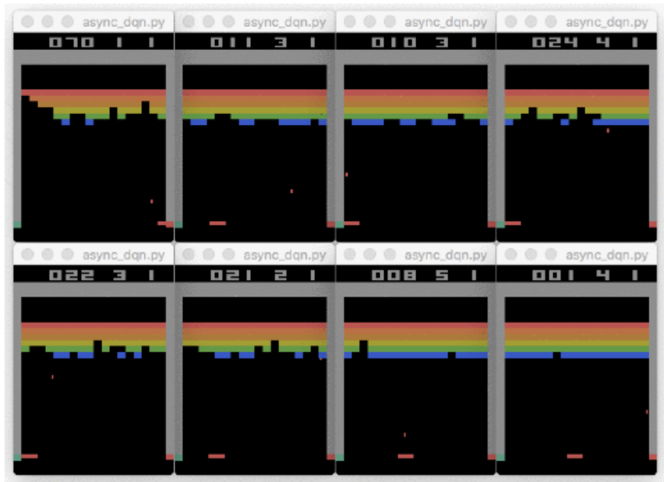
Adversarial Game



- Most decision making problems have more than one player.

All agents in the game aim at maximizing their own utilities.

Adversarial Game



- Most decision making problems have more than one player.

All agents in the game aim at maximizing their own utilities.

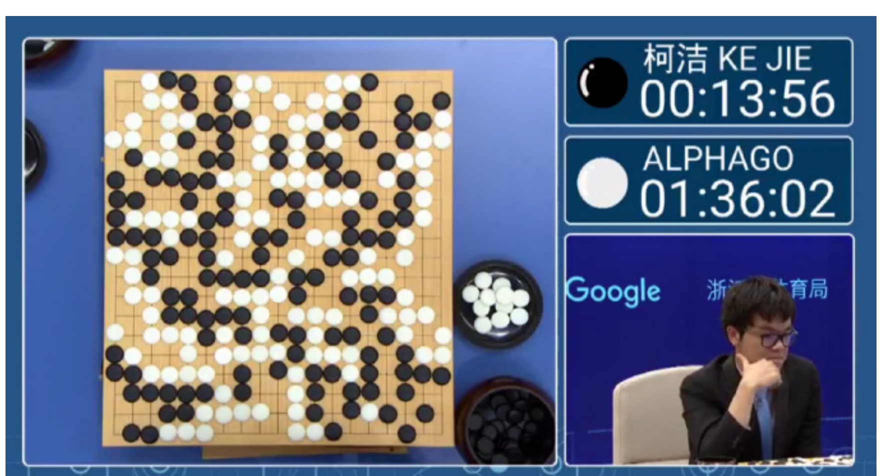
Types of Adversarial Games

- Nature of environment
  - Deterministic
  - Stochastic
- Availability of information
  - Complete
  - Imperfect, incomplete

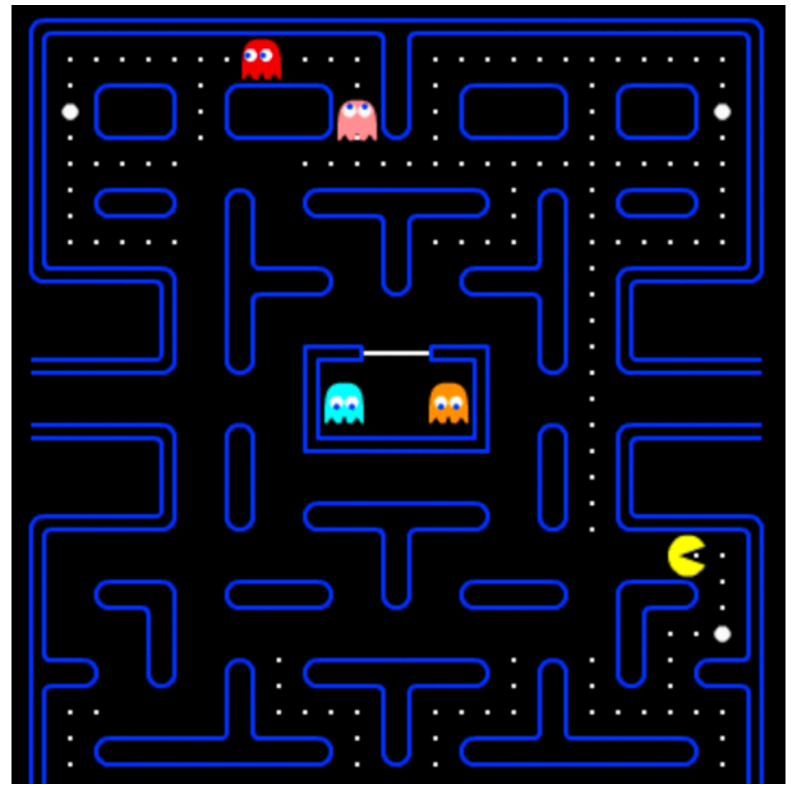
Types of Adversarial Games

- Nature of environment
  - Deterministic
  - Stochastic
- Availability of information
  - Complete
  - Imperfect, incomplete

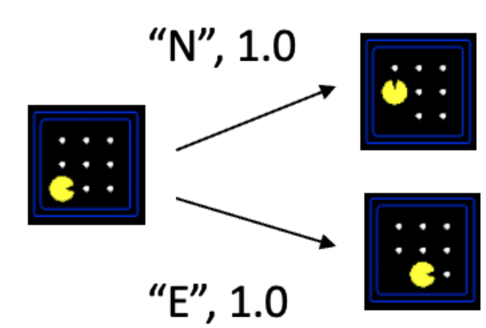
Currently, we consider only deterministic game with complete information.
We will start to deal with stochastic game later today.
We will start to deal with imperfection and incompleteness from the next lecture.



Deterministic & Complete Games

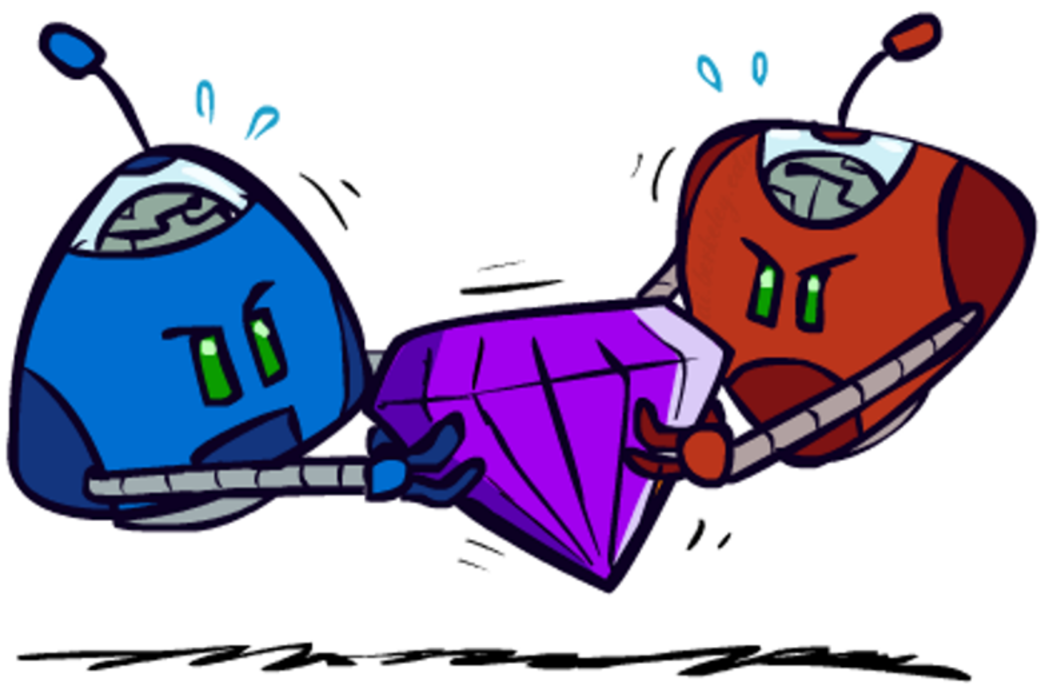


- A state space: 
- A transition function: state $\times$ action $\rightarrow$ (state, utility)
- A start state and goal test



All these are deterministic and known by the player!

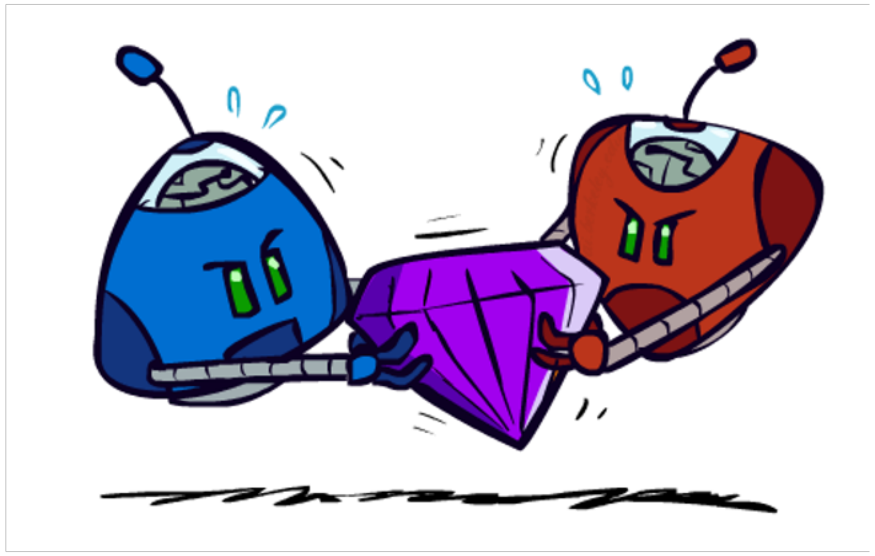
But now we have more than one player…



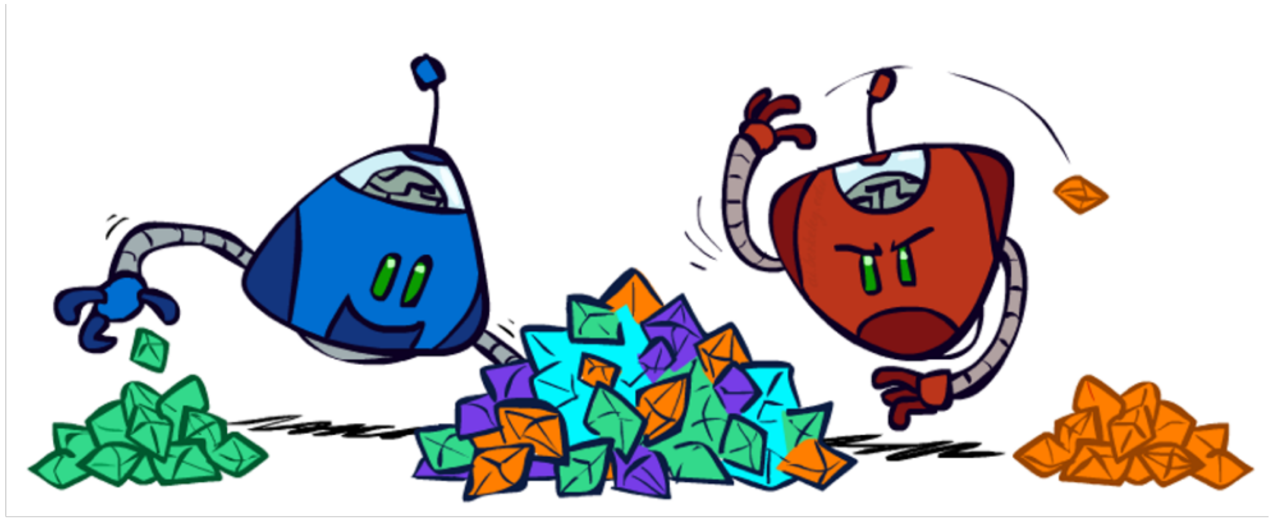
Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Two-Player Zero-Sum Game



vs.



In zero-sum games, the utility functions of the two players are coupled:
How much one wins equals to how much the other loses: competitive.
Can the players be cooperative?

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

- A MinMax Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

- A MinMax Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

- A MinMax Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

- A MinMax Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

- A MinMax Game.

3	5	2
6	8	4
7	10	9

$$U(i^*, j^*) = \min_i \max_j U(i, j)$$

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

$$U(i^*, j^*) = \max_j \min_i U(i, j)$$

Is MinMax equivalent to MaxMin?

3	5	2
6	8	4
7	10	9

Does Alice (Bob) lose (win) the same utility in MinMax and MaxMin games?

Is MinMax equivalent to MaxMin?

3	5	2
6	8	4
7	10	9

Does Alice (Bob) lose (win) the same utility in MinMax and MaxMin games?

Theorem:

$$\max_{P_j} \min_{P_i} \mathbb{E}_{P_i,P_j} U(i,j) \leq \min_{P_i} \max_{P_j} \mathbb{E}_{P_i,P_j} U(i,j)$$

Moving first is always worse!

Deterministic vs. Stochastic Strategy

	0.1	0.4	0.5
0.3	3	5	2
0.3	6	8	4
0.4	7	10	9

Mixed (Stochastic) strategy: choosing a distribution over actions instead of a single action (pure or deterministic strategy)

Deterministic vs. Stochastic Strategy

$$\begin{array}{c|ccc|}
 & 0.1 & 0.4 & 0.5 \\
\hline
0.3 & 3 & 5 & 2 \\
0.3 & 6 & 8 & 4 \\
0.4 & 7 & 10 & 9 \\
\hline
\end{array}$$

Mixed (Stochastic) strategy: choosing a distribution over actions instead of a single action (pure or deterministic strategy)

Von Neumann's Minimax Theorem:

$$\min_{P_i} \max_{P_j} \mathbb{E}_{P_i, P_j} \left[U(i, j) \right] = \max_{P_j} \min_{P_i} \mathbb{E}_{P_i, P_j} \left[U(i, j) \right]$$

MinMax and MaxMin Games are equivalent for mixed strategy!

Nash Equilibrium

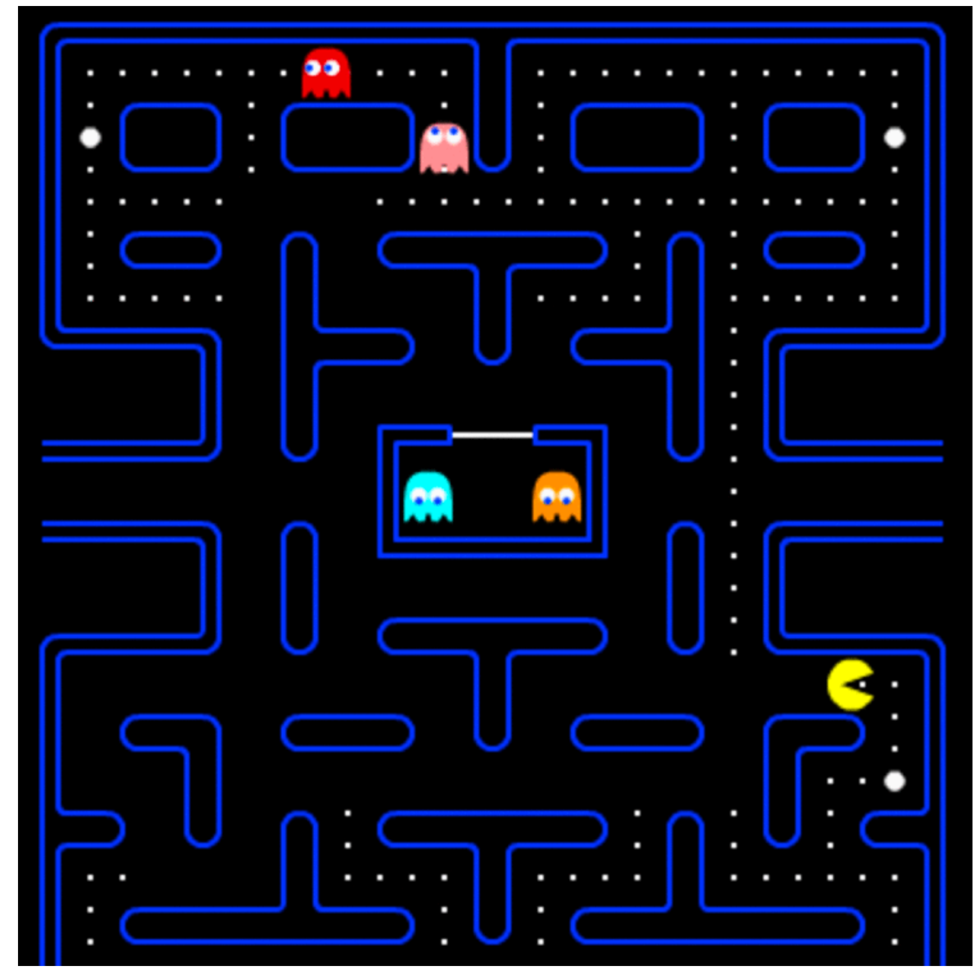
$$\min_{P_i} \max_{P_j} \mathbb{E}_{P_i, P_j} \left[U(i,j) \right] = \max_{P_j} \min_{P_i} \mathbb{E}_{P_i, P_j} \left[U(i,j) \right]$$

- Strong duality: the solution pair P_i^*, P_j^* achieving this equation is called the saddle point of the two-step zero-sum game.
- This is also the Nash equilibrium of the game.

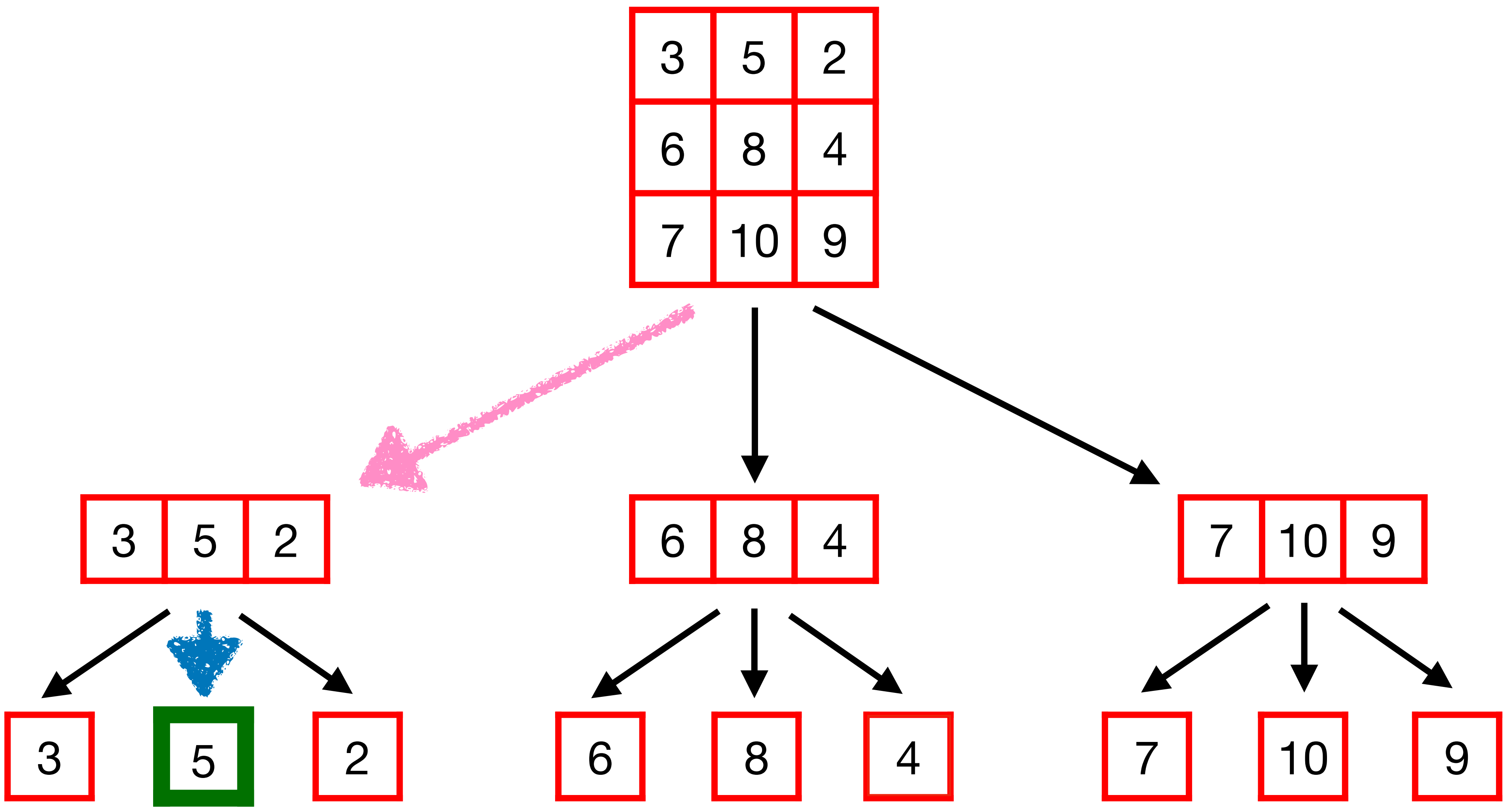
Under Nash equilibrium, changing the strategy for any player herself would not be a good idea.
But Nash equilibrium does not mean optimal strategy!
You know the prisoner's dilemma.

Multi-Step Zero-Sum Game

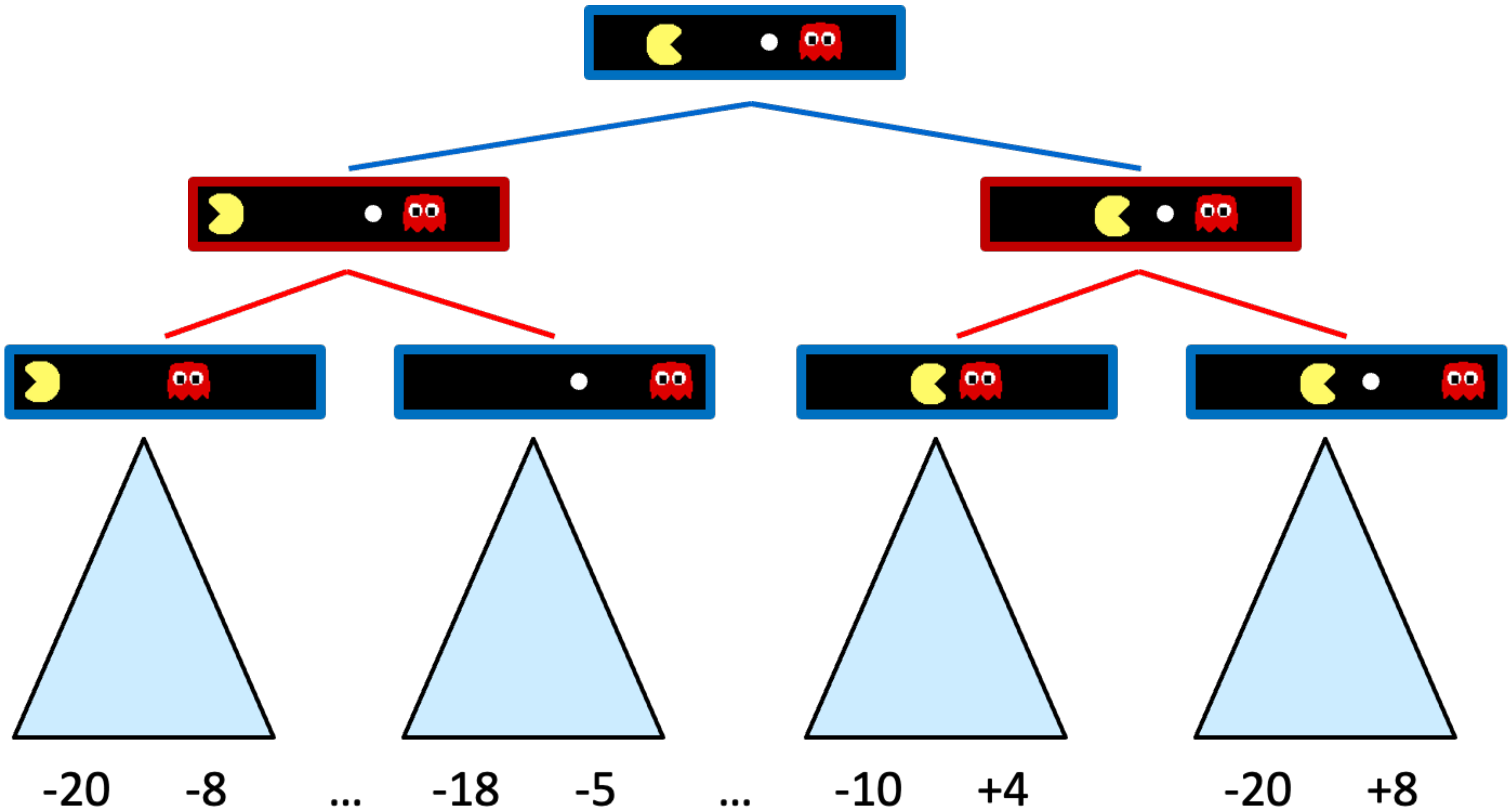
3	5	2
6	8	4
7	10	9



Game Tree of Two-Step Games



Multi-Step Zero-Sum Game



Can also be modeled as a search tree.

Lesson learned from two-step game:
The players should play rationally: use min and max strategies.
The players should look ahead when making decisions.

Min and Max Strategy

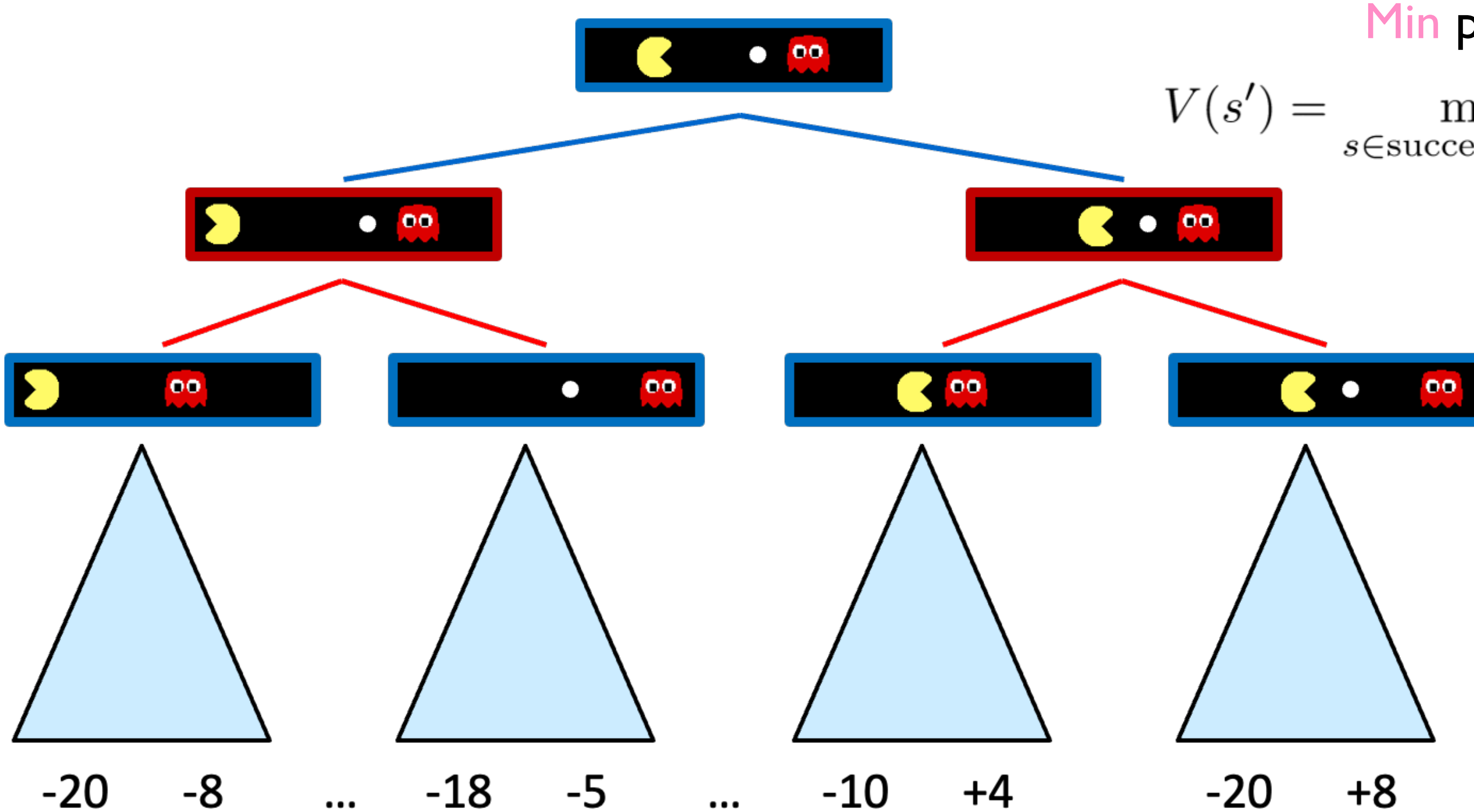
The game is deterministic and complete:
The search tree is known to the players.

Max player

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

Min player

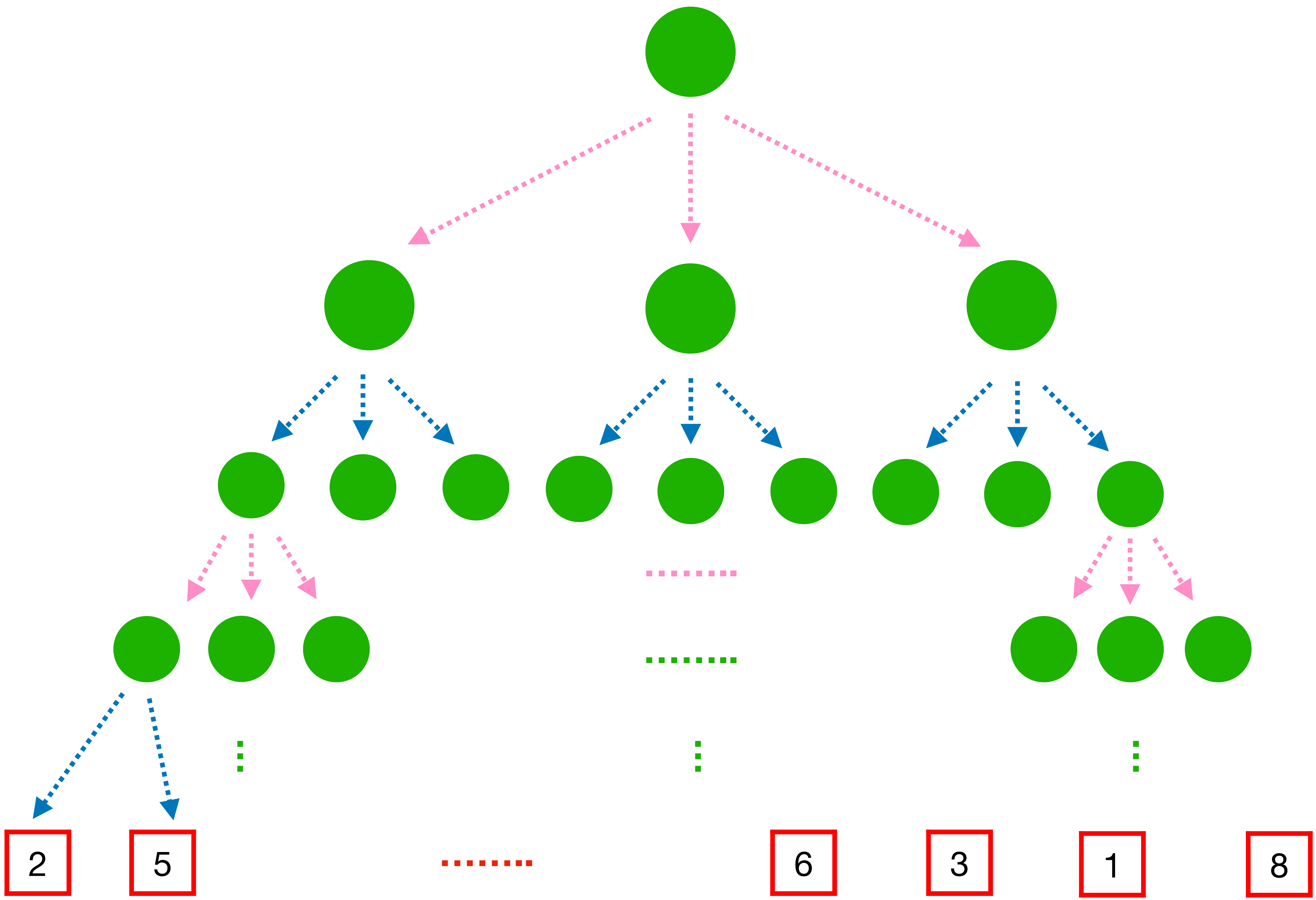
$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$



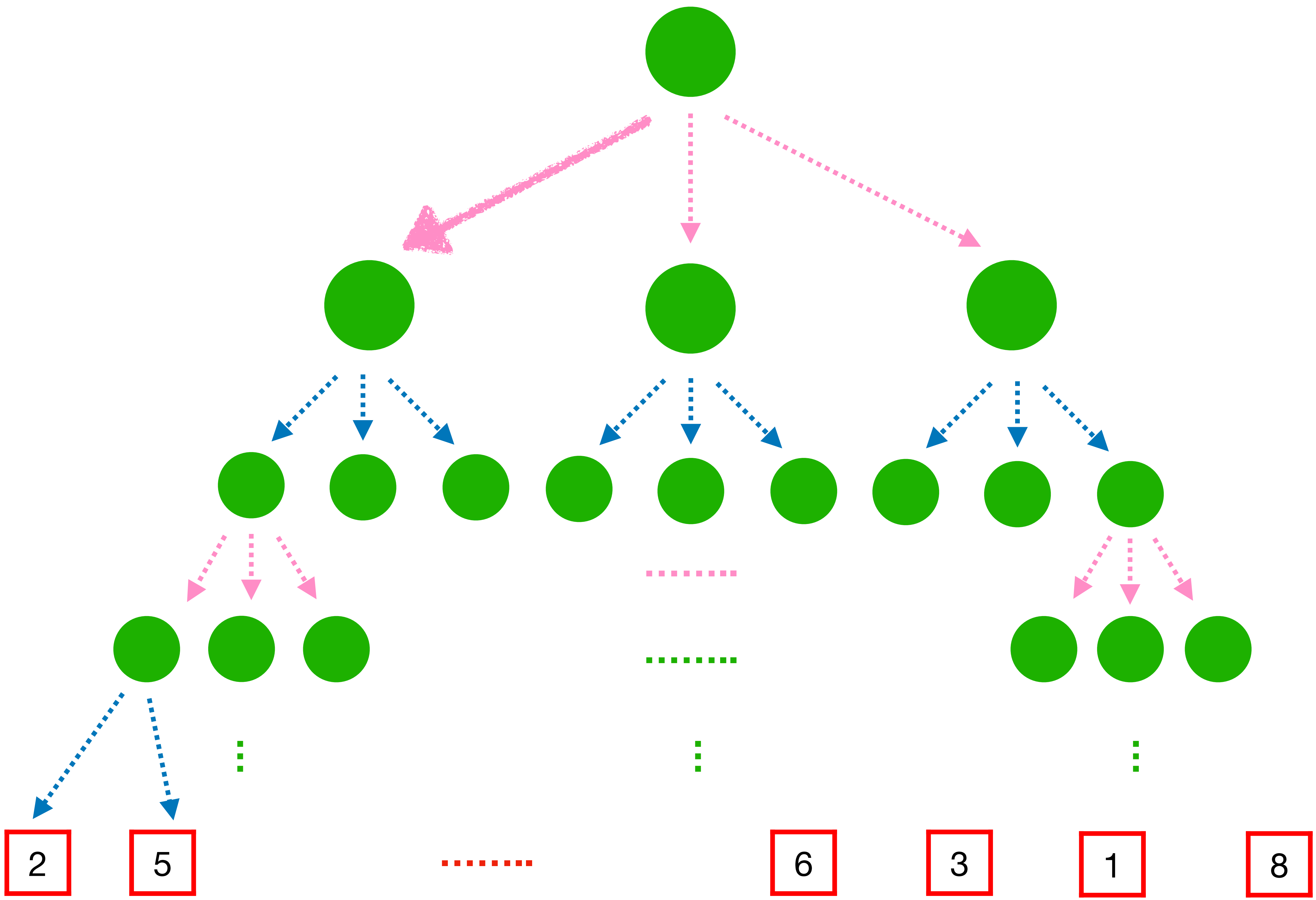
Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

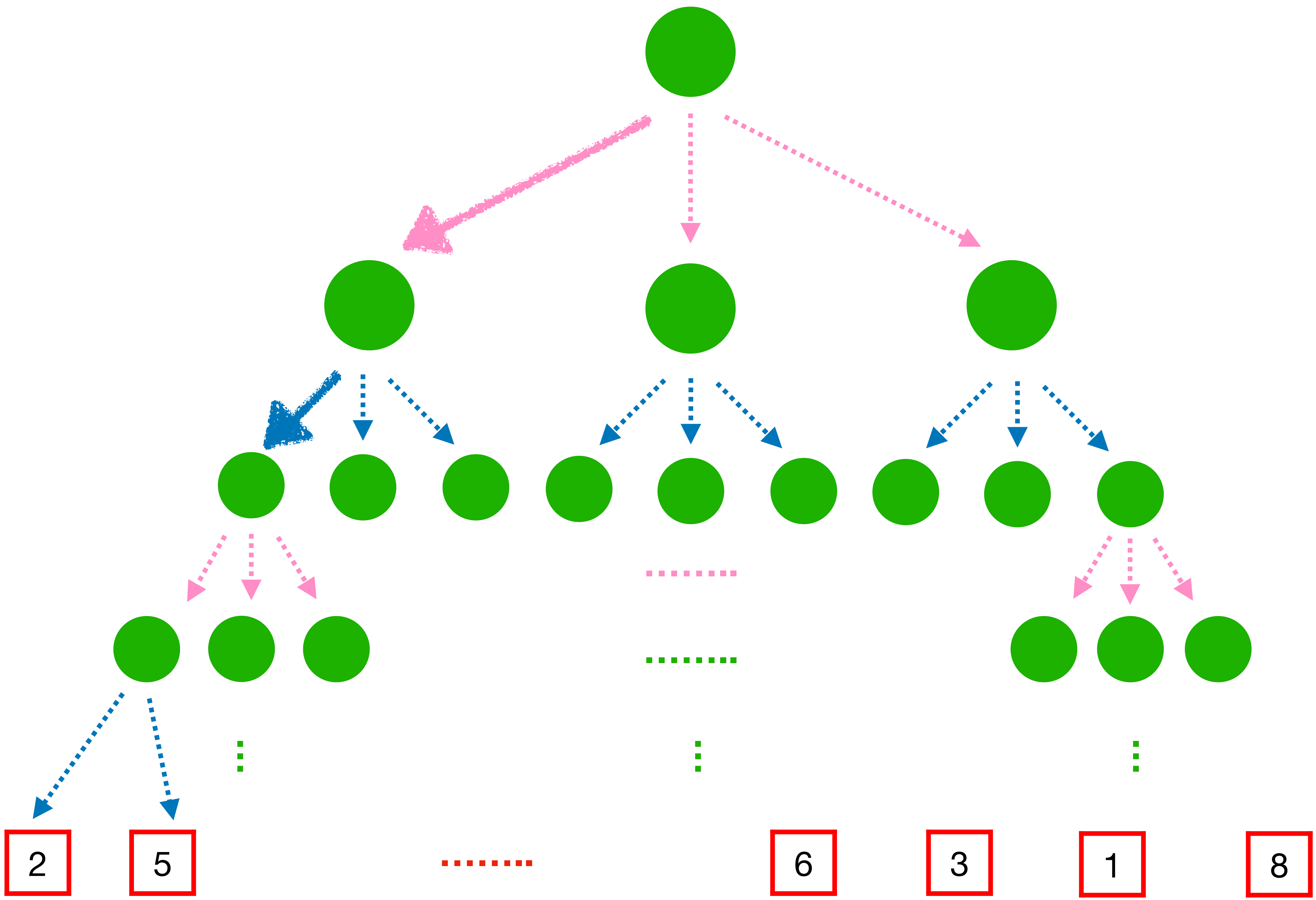
Minimax Search



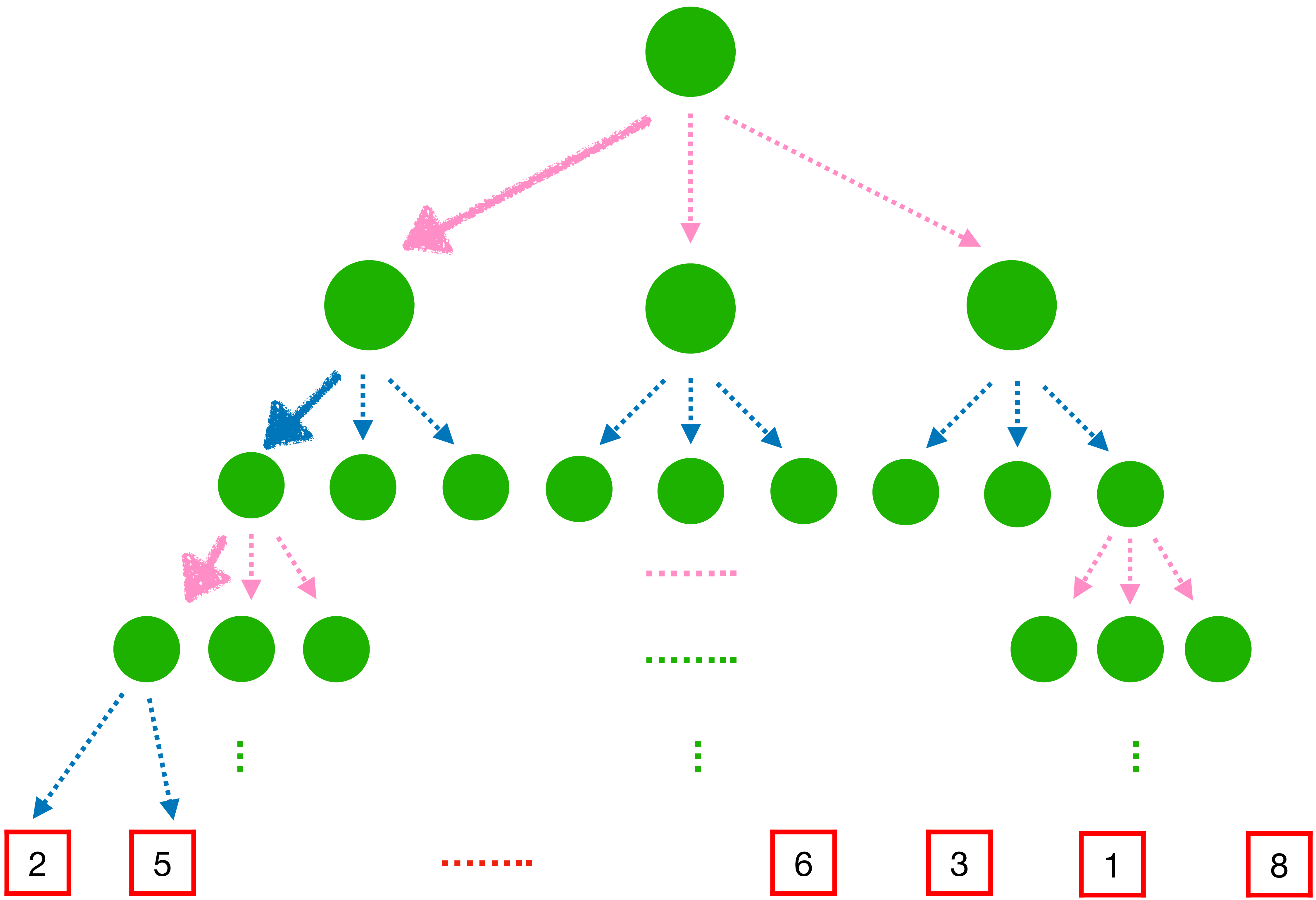
Minimax Search



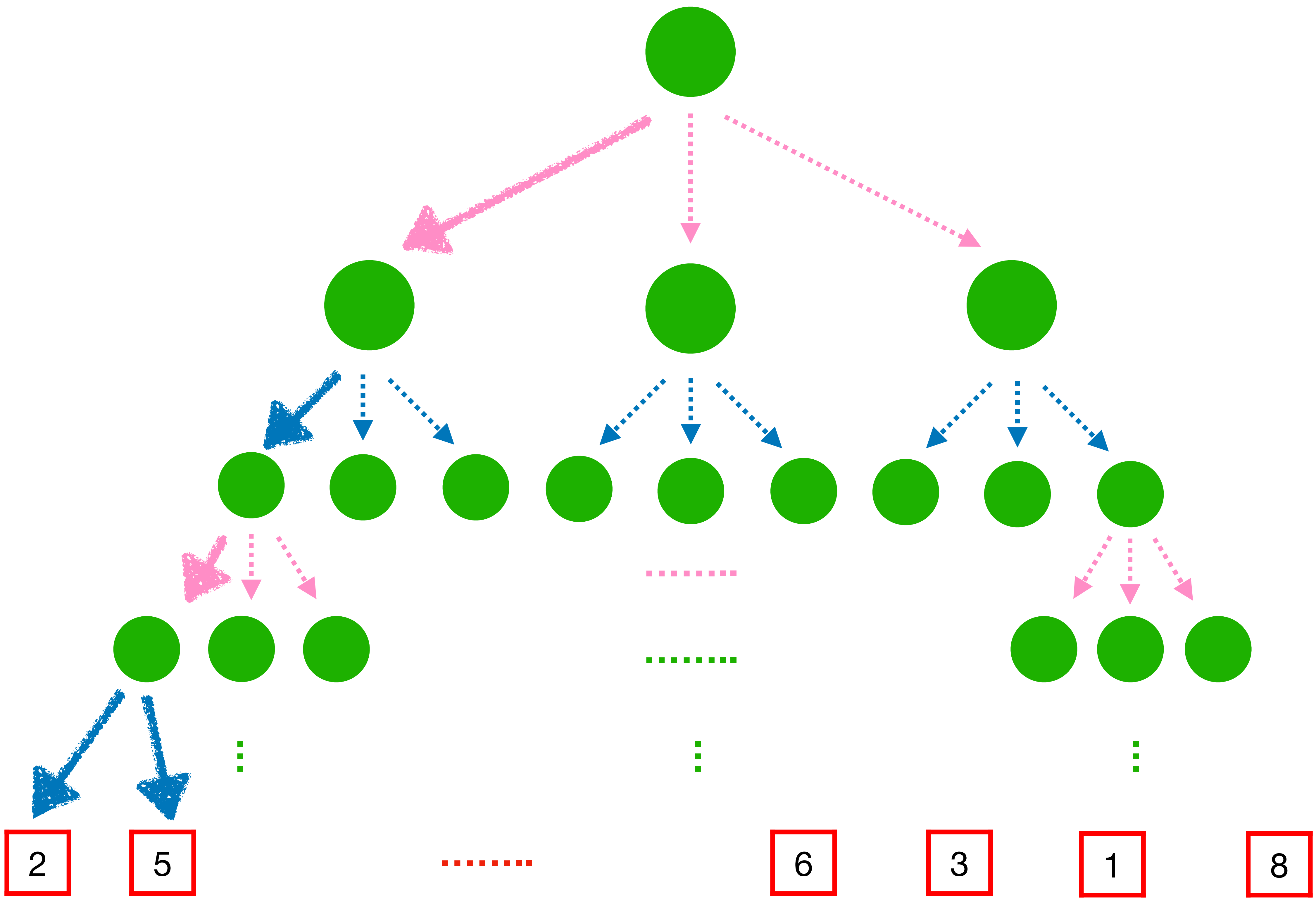
Minimax Search



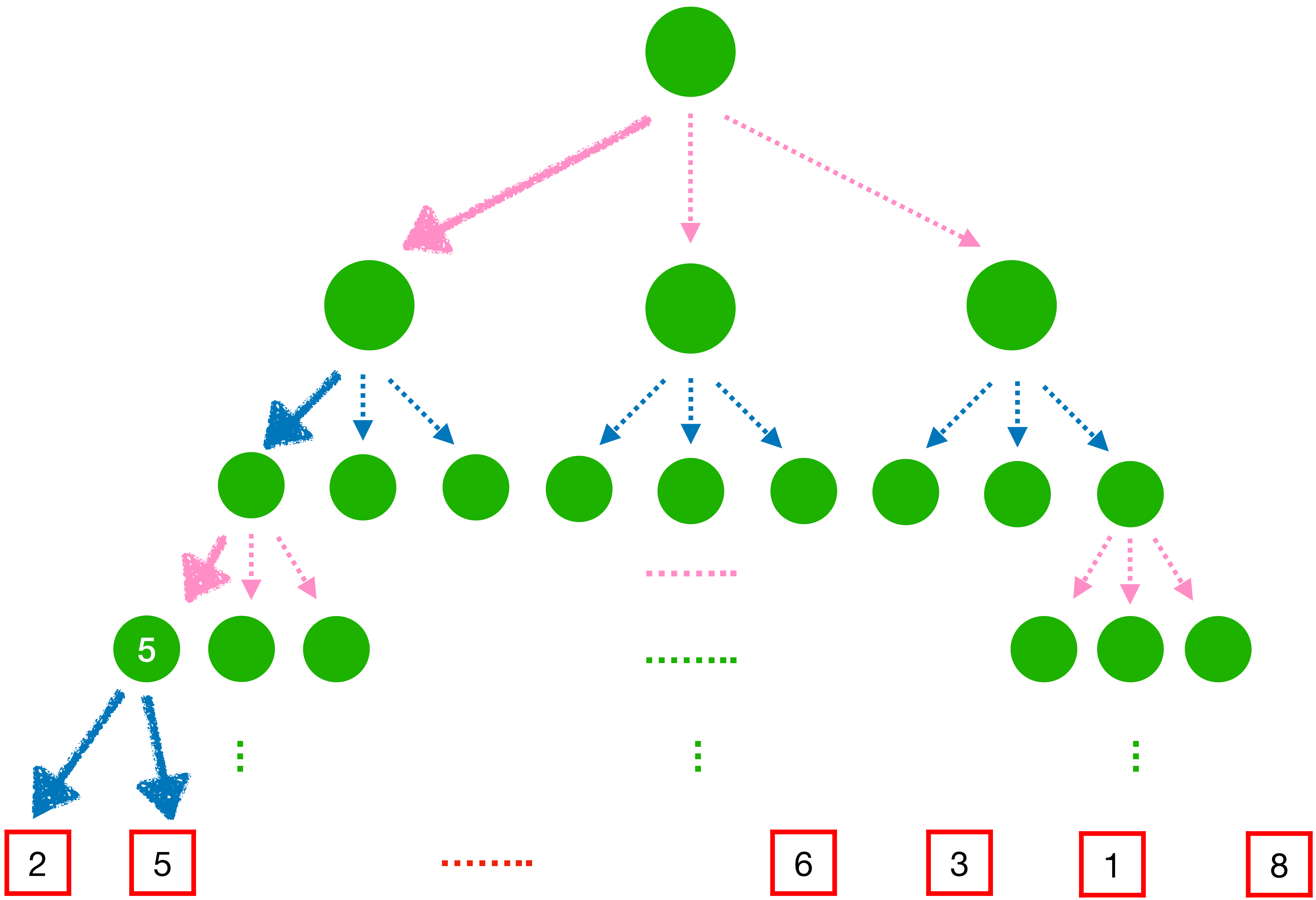
Minimax Search



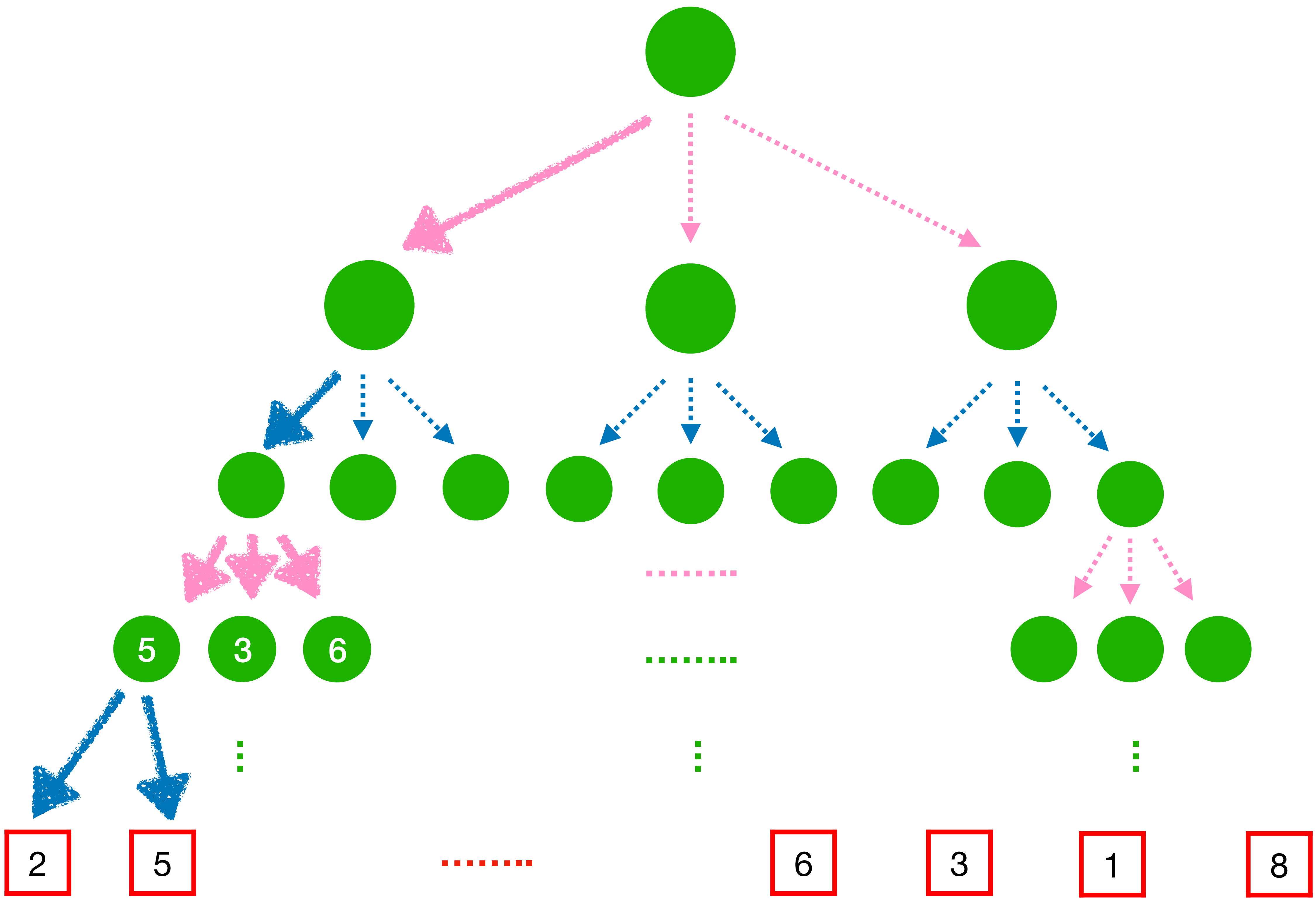
Minimax Search



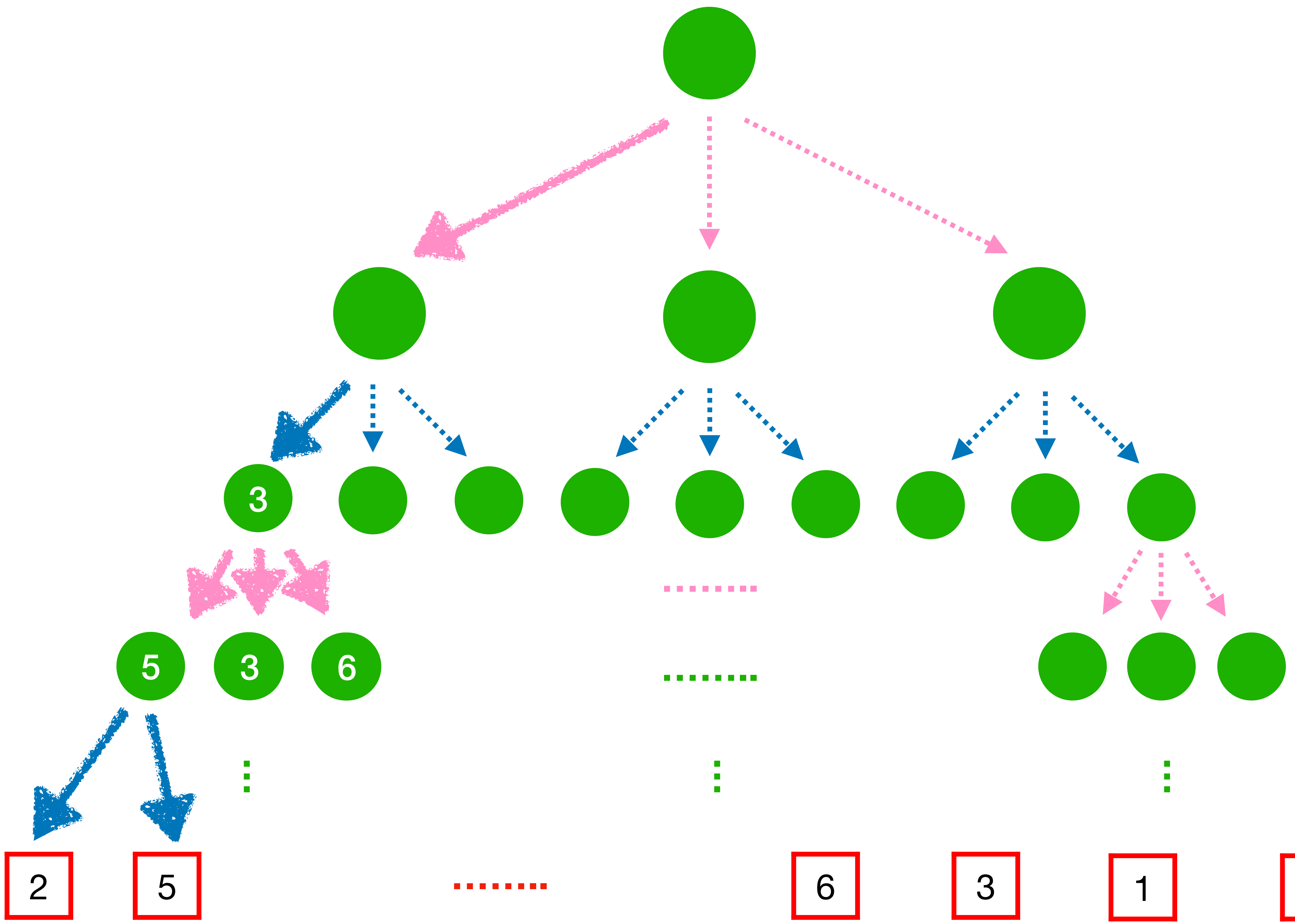
Minimax Search



Minimax Search



Minimax Search



Minimax Search

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

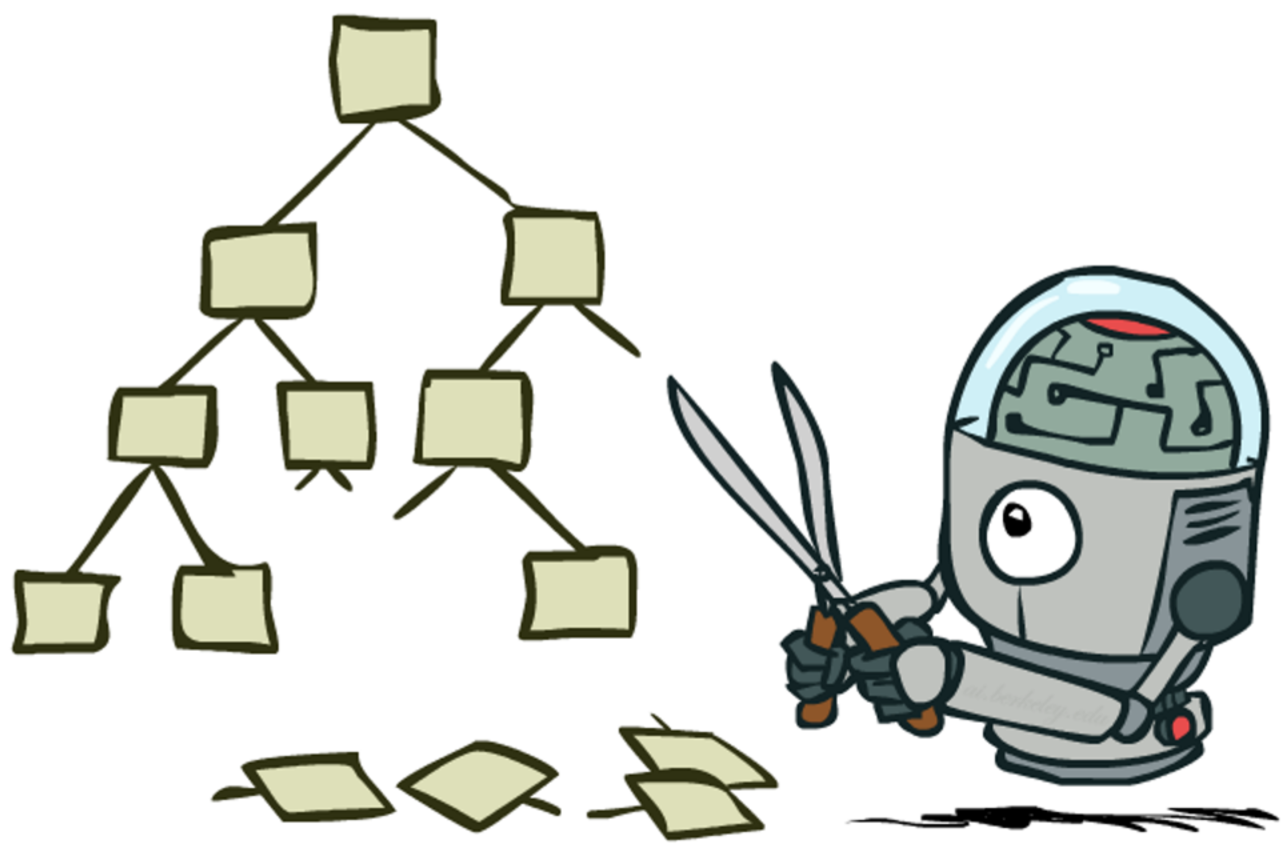
```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

Works quite similar to depth-first search:
Search down then trace back
Similar time and space complexity to DFS

Can We Search More Efficiently?

- Limit the search depth and use heuristic functions to estimate the finally utility: e.g. how far away is the ghost?

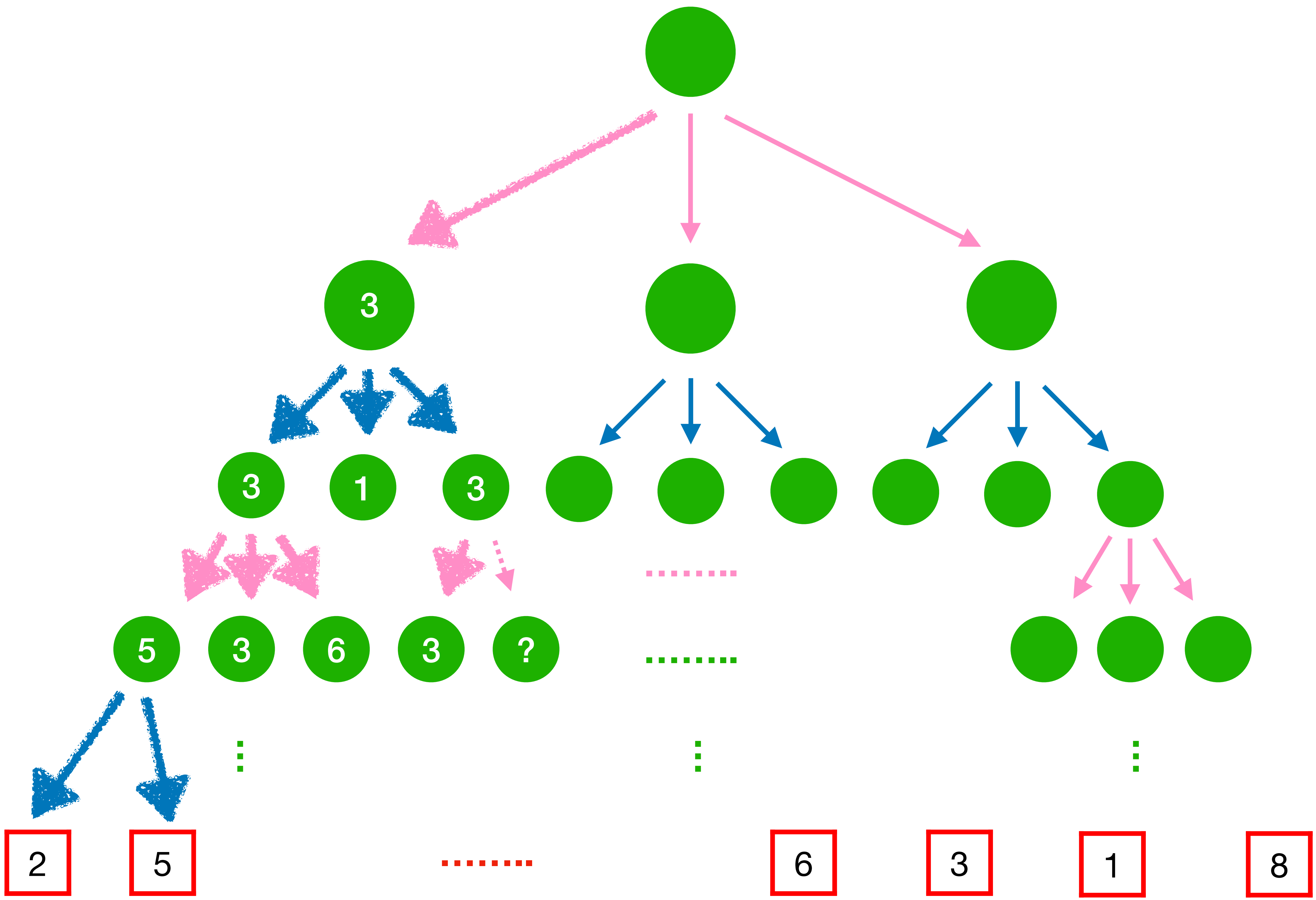
- Prune the search tree.



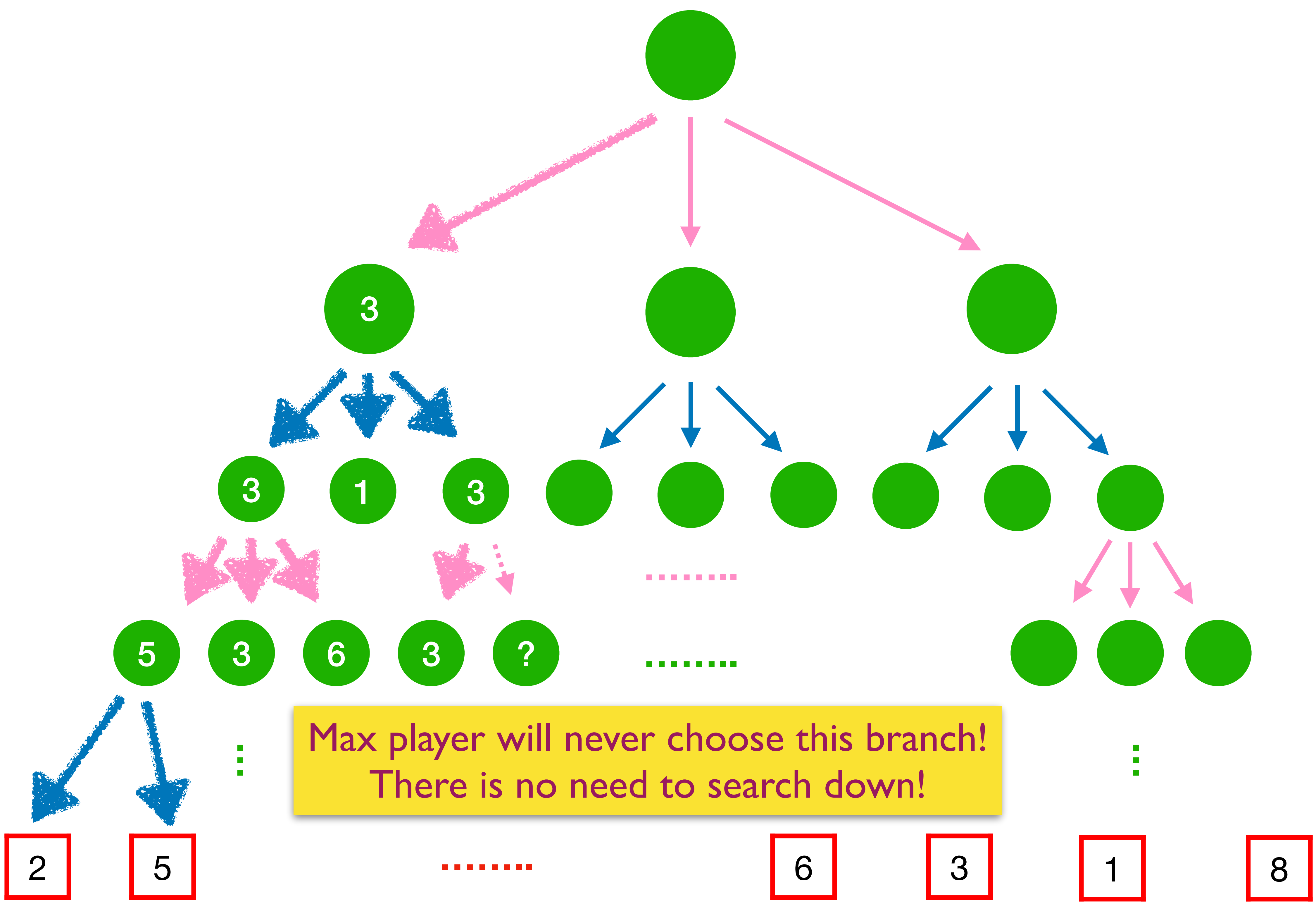
Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Alpha-Beta Pruning

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

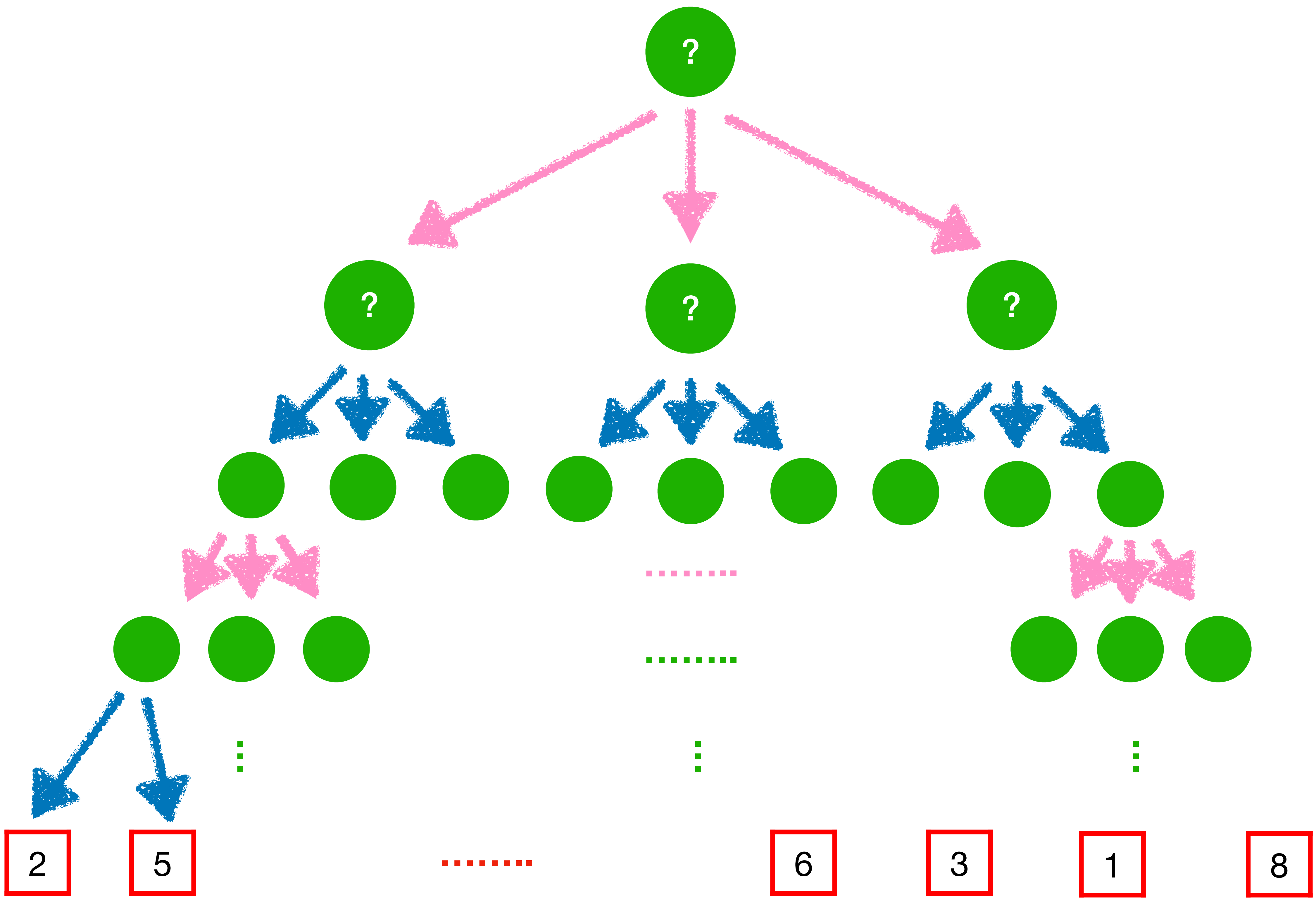
```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Search order matters!
In the worst case, no pruning will be done with bad order!
Deterministic search is not aware of tree structure.
Can we search more smartly?

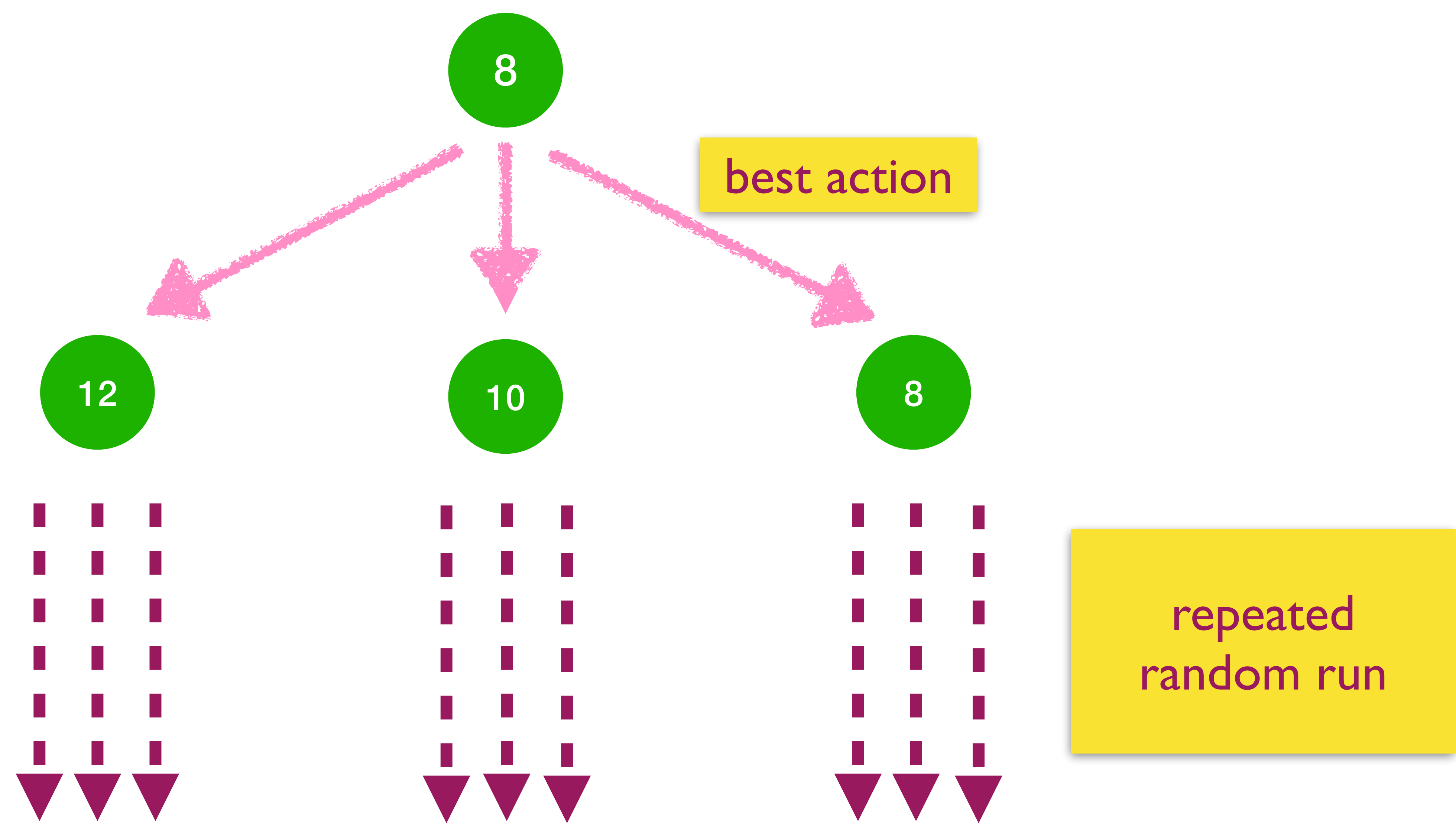
Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Suppose that the search tree is fully expanded, how to evaluate each mode?

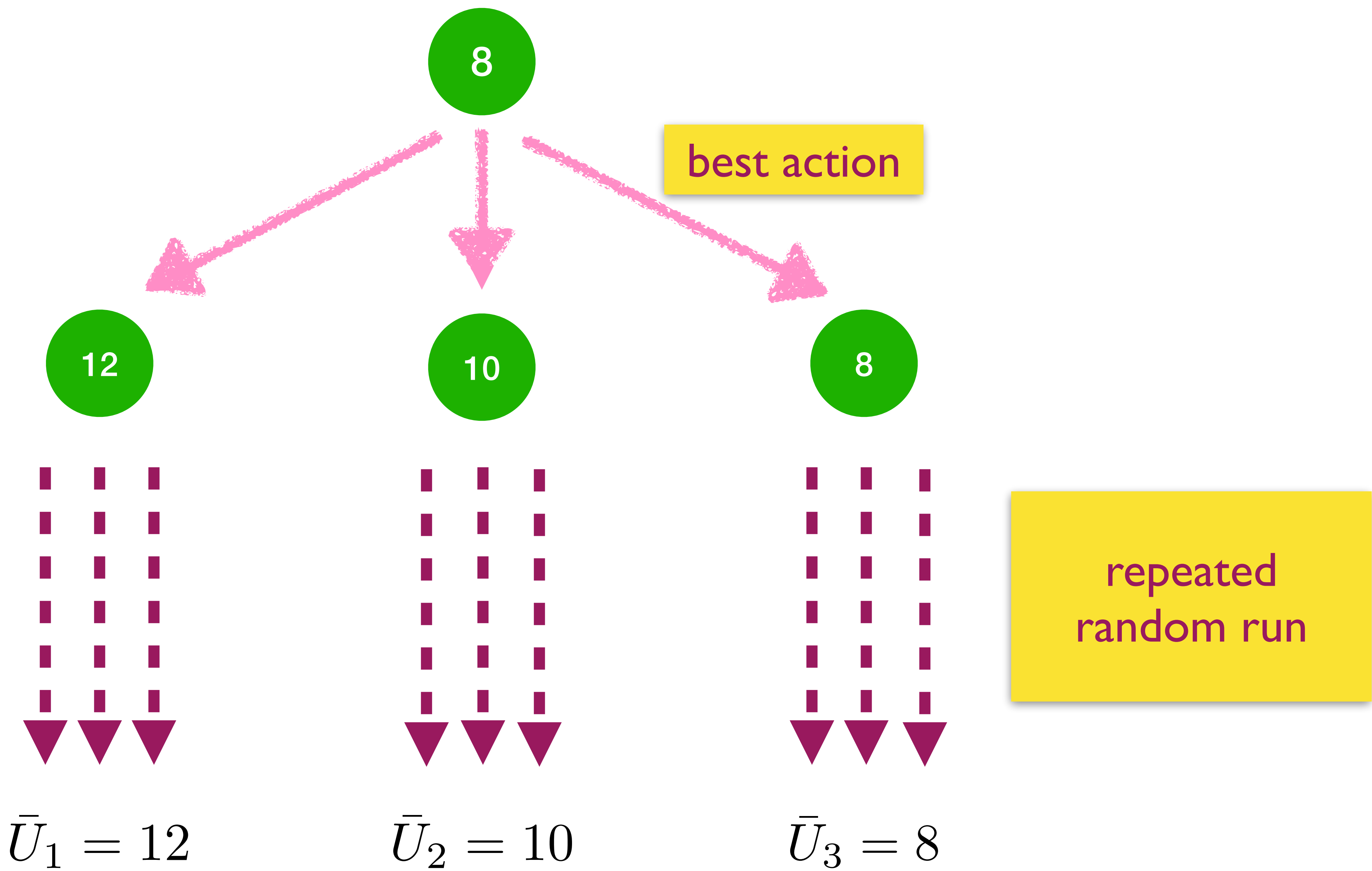


Monte-Carlo Simulation



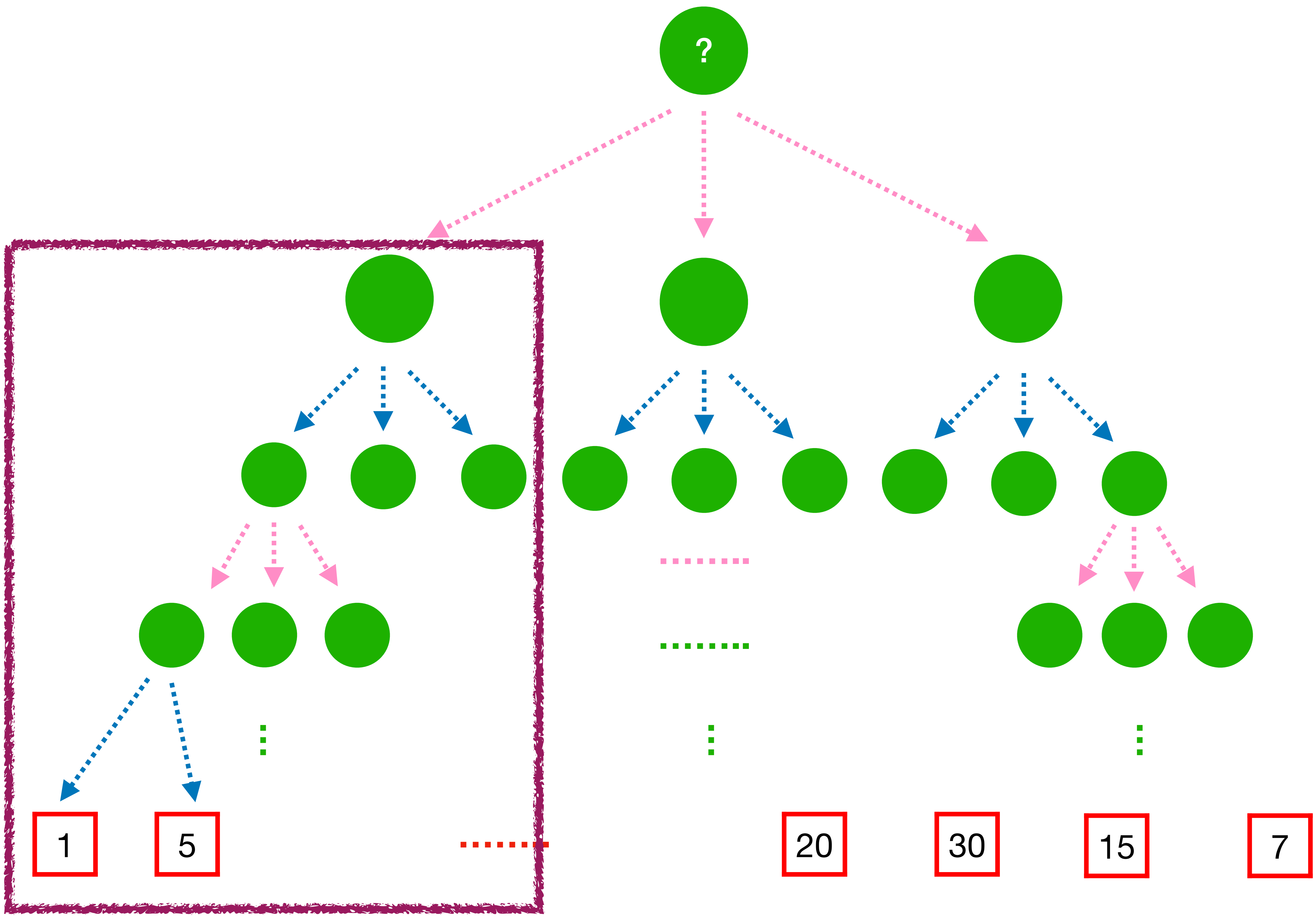
$$\bar{U}_1 = 12 \qquad \bar{U}_2 = 10 \qquad \bar{U}_3 = 8$$

Monte-Carlo Simulation

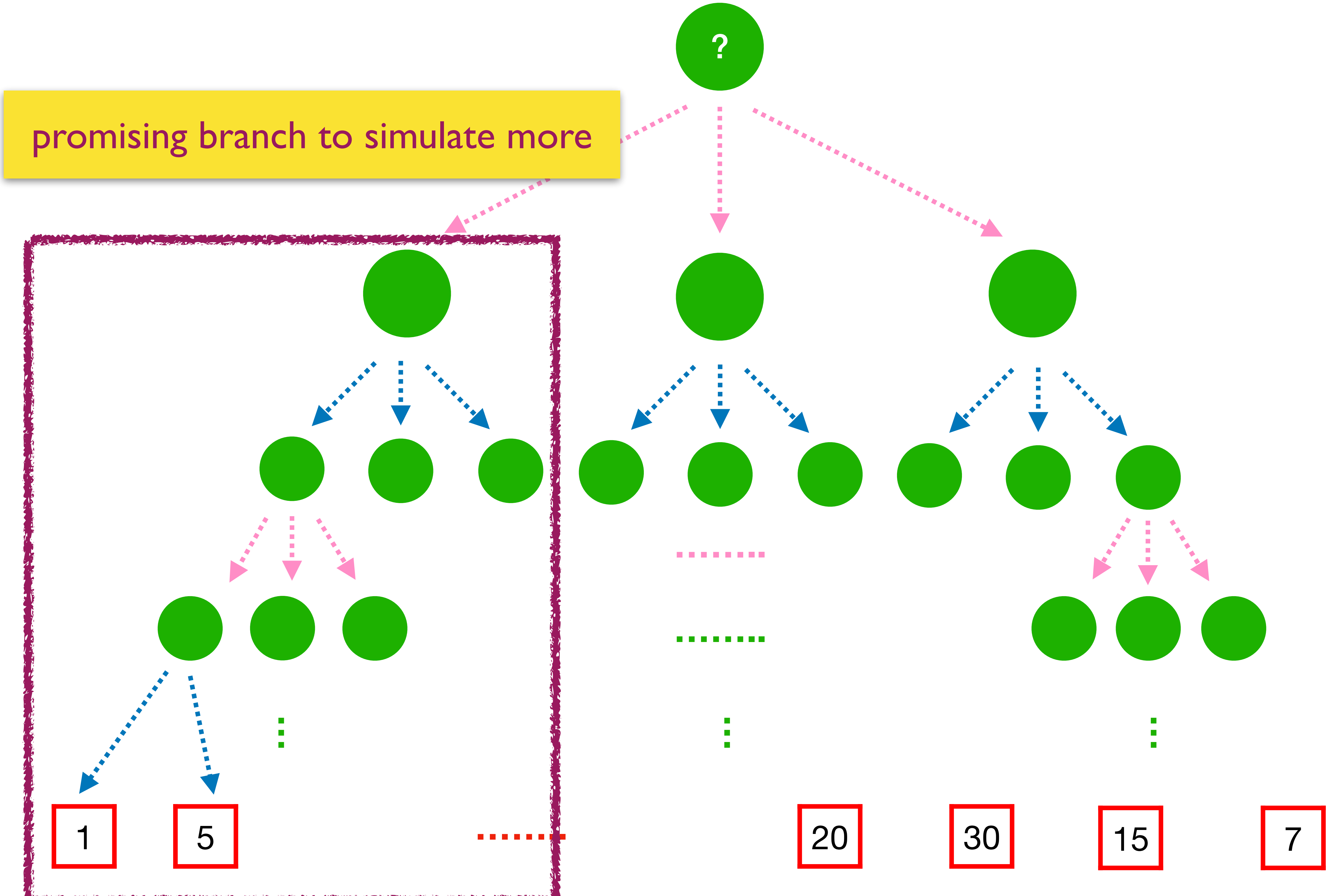


Converge to true utility when the #run is sufficient!
But in real game playing, the time and space for simulation is limited.
We need a smart strategy to decide the order of simulation.

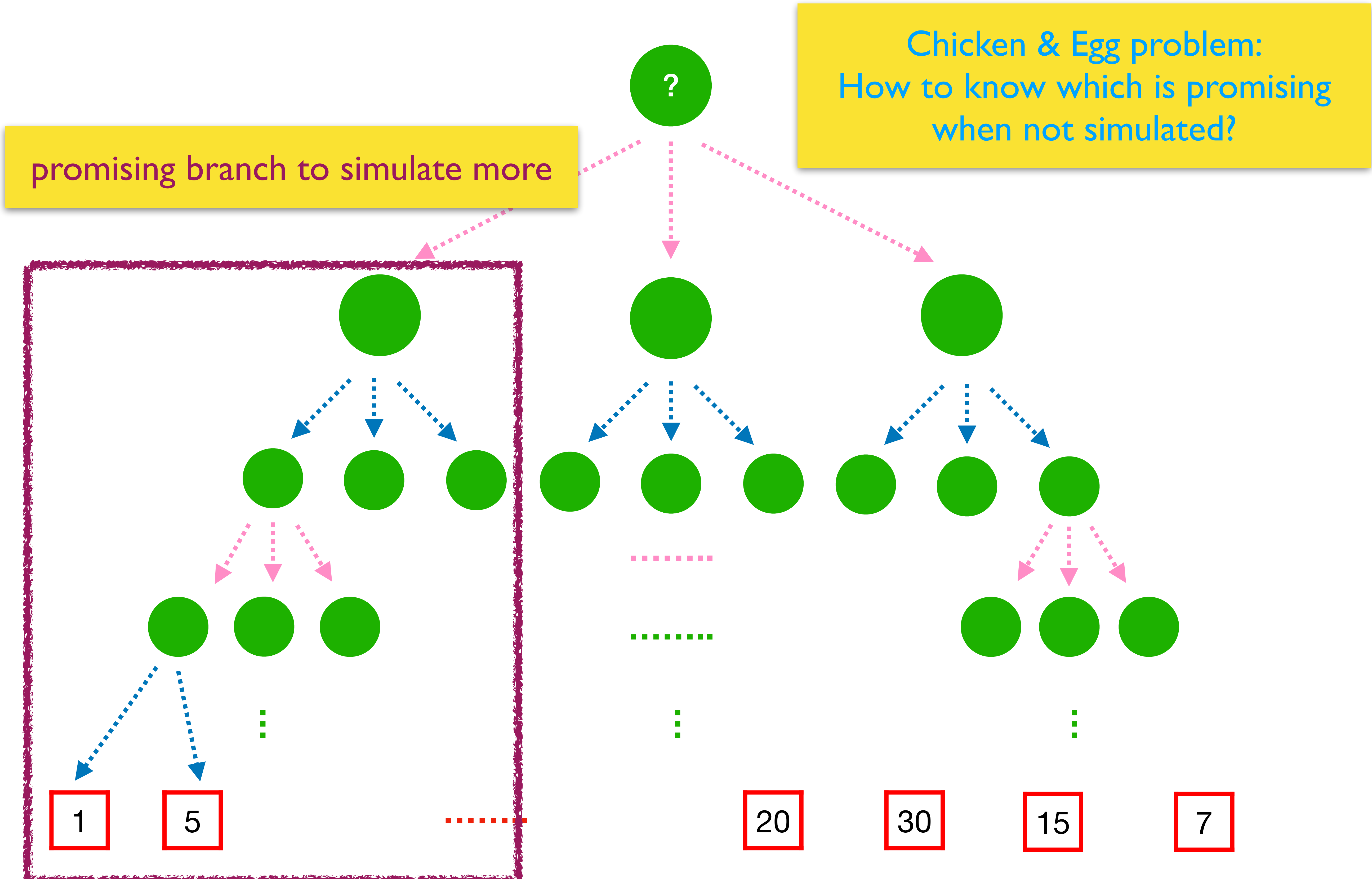
The Order of Simulation Matters



The Order of Simulation Matters

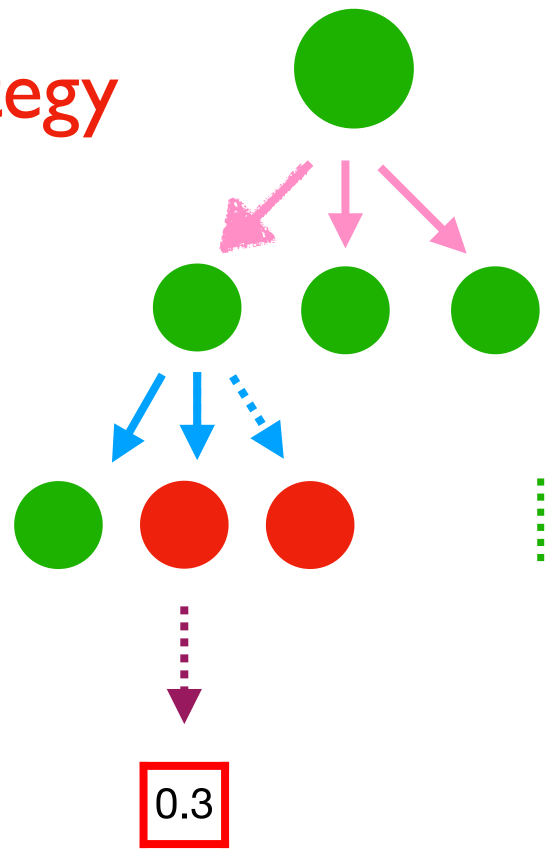


The Order of Simulation Matters



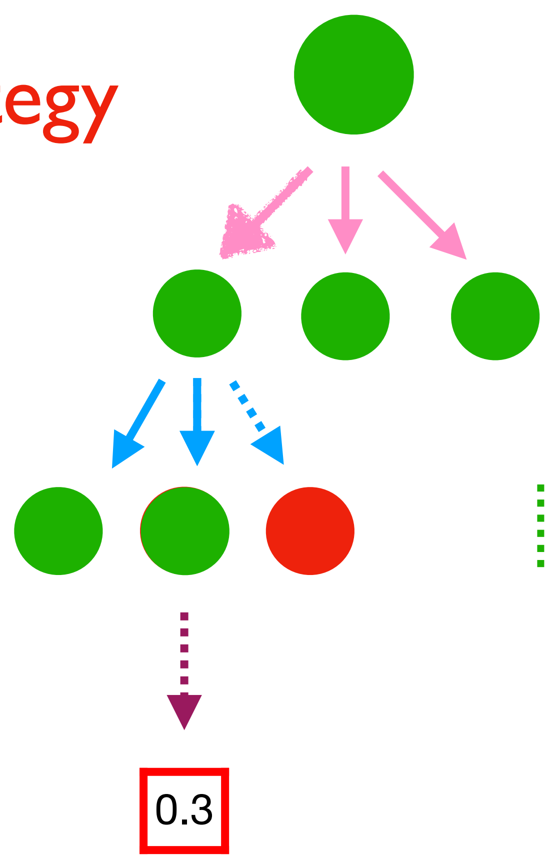
Monte-Carlo Tree Search (MCTS)

- Nodes of two kinds: ● visited node, ● unvisited node.
- During MCTS, we use two strategies: tree and default
- Algorithm: repeat until time or space limit:
  - Selection: choose one node among ● using tree strategy
  - Expansion: if the node has unvisited child, expand one and put into ●
  - Simulation: simulate down using default strategy
  - Update: update MC estimation through path
- Output the best action to play



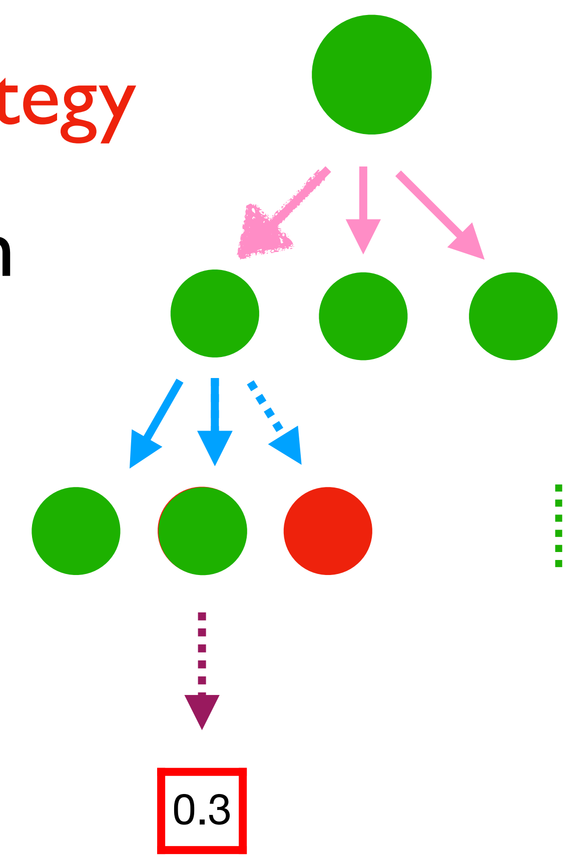
Monte-Carlo Tree Search (MCTS)

- Nodes of two kinds: visited node, unvisited node.
- During MCTS, we use two strategies: tree and default
- Algorithm: repeat until time or space limit:
  - Selection: choose one node among using tree strategy
  - Expansion: if the node has unvisited child, expand one and put into
  - Simulation: simulate down using default strategy
  - Update: update MC estimation through path
- Output the best action to play



Monte-Carlo Tree Search (MCTS)

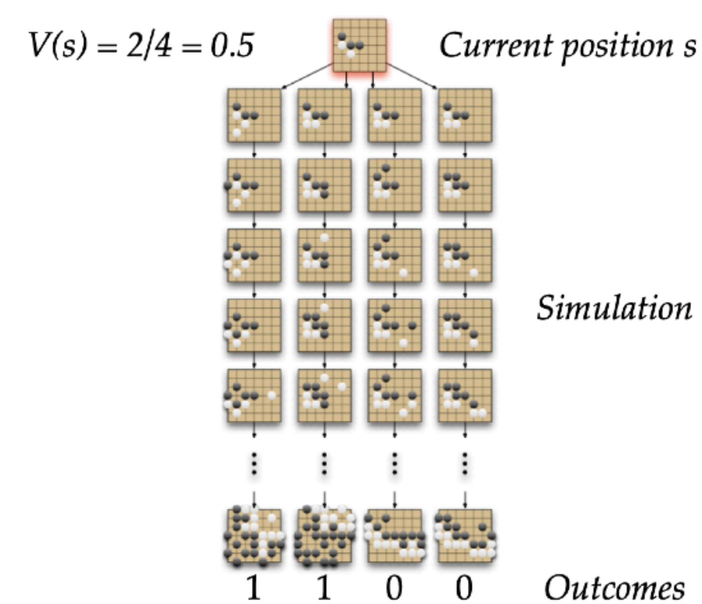
- Nodes of two kinds: visited node, unvisited node.
- During MCTS, we use two strategies: tree and default
- Algorithm: repeat until time or space limit:
  - Selection: choose one node among using tree strategy
  - Expansion: if the node has unvisited child, expand one and put into
  - Simulation: simulate down using default strategy
  - Update: update MC estimation through path
- Output the best action to play



The default strategy is usually random play
The tree strategy is essential:
Deciding the order of search

Monte-Carlo Tree Search (MCTS)

- Free to choose tree and default strategy
  - Combine with other game-play strategy, e.g. reinforcement learning
- Stop in anytime, capable for real-time play
- Multiple simulation can be done in parallel
- Can run in stochastic environment
  - e.g. adversary uses mixed strategy
  - state transition is stochastic
- Limitation: still needs to know the model (environment)!



Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing *K* arms, and each time he pulls 1 arm and receives a reward

    Arm 1
    Arm 2
    Arm 3

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$
Arm 2	10
Arm 3	$X_{3,1}$

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$
Arm 2	10	$X_{2,2}$
Arm 3	$X_{3,1}$	0

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- Stochastic setting: the rewards are sampled from unknown stochastic distributions
- Adversarial setting: the rewards are chosen by an adversary
- Goal: Maximizing cumulative rewards or Finding the best arm

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- Stochastic setting: the rewards are sampled from unknown stochastic distributions

- Adversarial setting: the rewards are chosen by an adversary

- Goal: Maximizing cumulative rewards or Finding the best arm

Multi-Armed Bandits

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- Stochastic setting: the rewards are sampled from unknown stochastic distributions
- Adversarial setting: the rewards are chosen by an adversary
- Goal: Maximizing cumulative rewards or Finding the best arm

One of the major areas in theoretical machine learning

Greedy Strategy for MAB

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- Consider the greedy strategy:
  - Maintain the mean rewards obtained from each arm
  - Choose the arm with largest mean reward in each round

Greedy Strategy for MAB

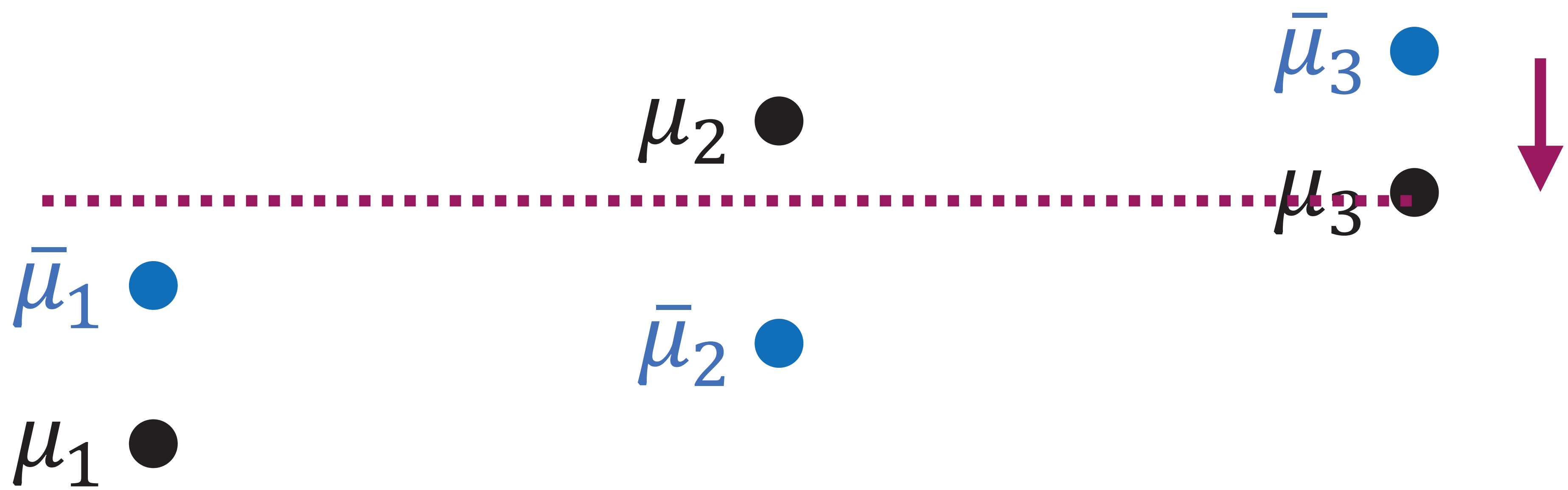
- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- Consider the greedy strategy:
  - Maintain the mean rewards obtained from each arm
  - Choose the arm with largest mean reward in each round

Will this strategy work?
No difference to random guess in the worst case :-(

Greedy Strategy for MAB



Greedy strategy will stuck in this situation

Exploration-Exploitation Trade-Off

- Multi-Armed Bandits (MAB)
  - A gambler is facing K arms, and each time he pulls 1 arm and receives a reward

Arm 1	$X_{1,1}$	$X_{1,2}$	6	$X_{1,4}$	$X_{1,5}$
Arm 2	10	$X_{2,2}$	$X_{2,3}$	0	$X_{2,5}$
Arm 3	$X_{3,1}$	0	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$

- The central challenges:
  - Environment is unknown and stochastic, we cannot know the reward if we don't pull an arm.
  - Data are collected by the player, not given by the environment, she could misled by herself!

The player should collect the data smartly:
prefer both good and uncertain arms!

Strategies Balancing Exploration and Exploitation

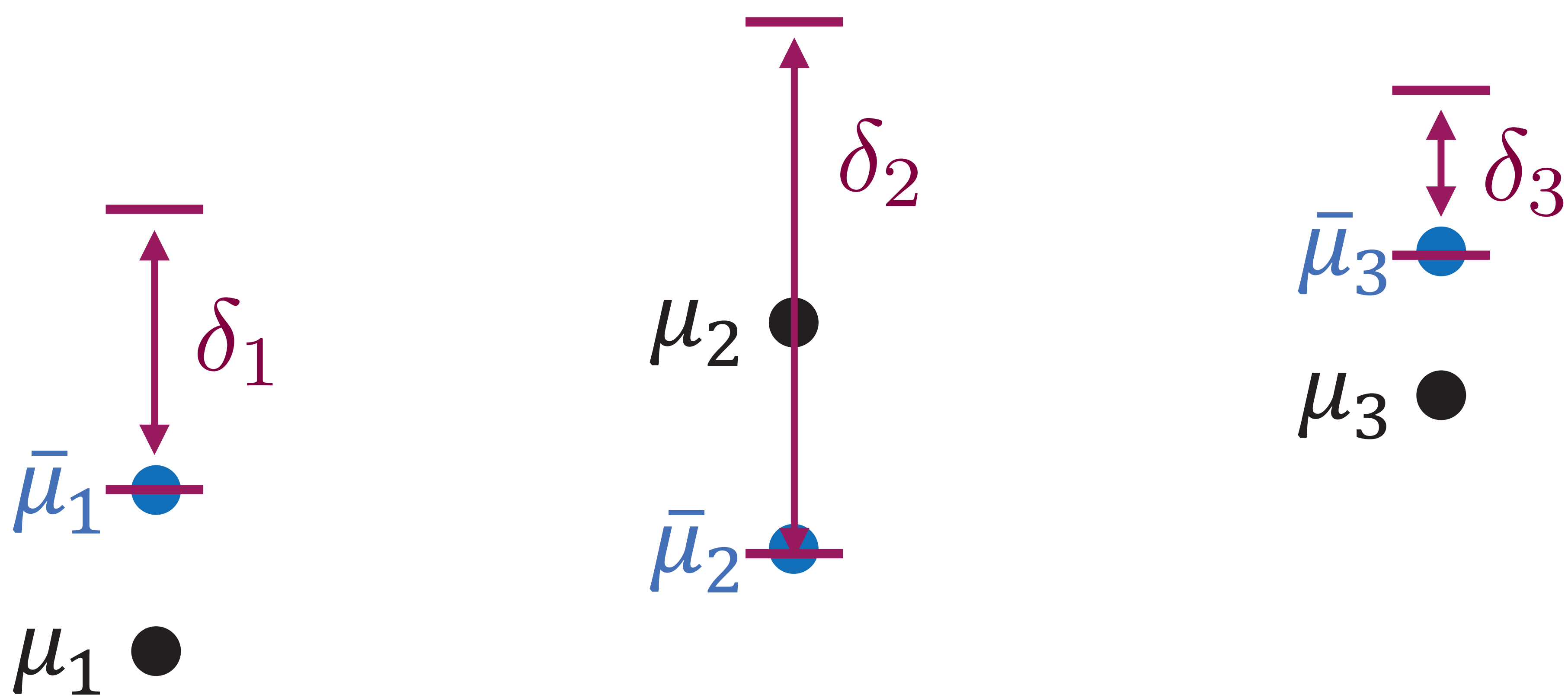
- ϵ-greedy strategy: with a decreasing probability of ϵ, the player chooses random arm, otherwise use greedy strategy.

- Thompson sampling:
  - maintain a posterior distribution of rewards for arms
  - Choose arms by sampling from this posterior distribution

Both are very popular strategy in practice!
While for theoretical optimality, we usually refer to the upper confidence bound strategy.

Upper Confidence Bound Strategy

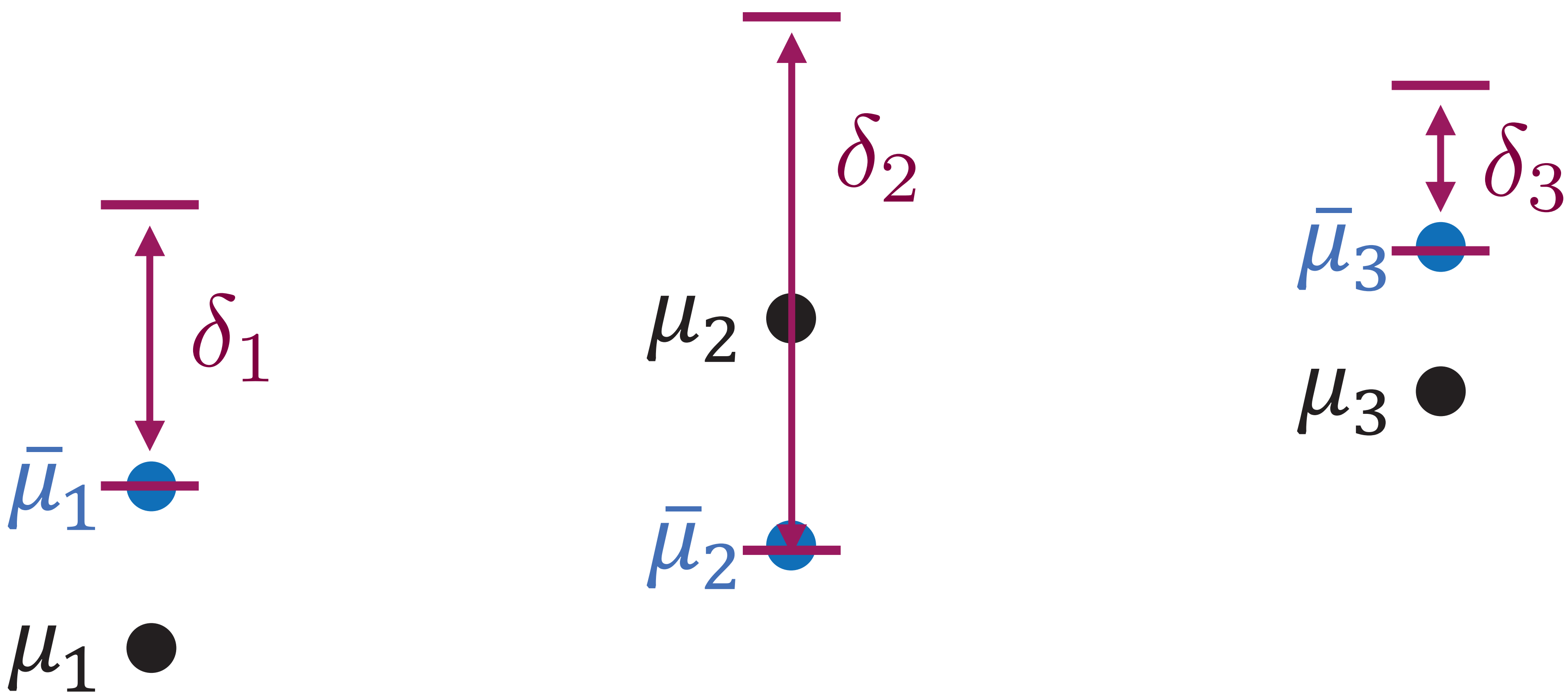
Choose the arm with the largest $\bar{\mu} + \delta$



- δ is the width of the confidence interval
- The width is calculated to ensure that $\mu \leq \bar{\mu} + \delta$ and the width decreases when the #pulls get large.

Upper Confidence Bound Strategy

Choose the arm with the largest $\bar{\mu} + \delta$

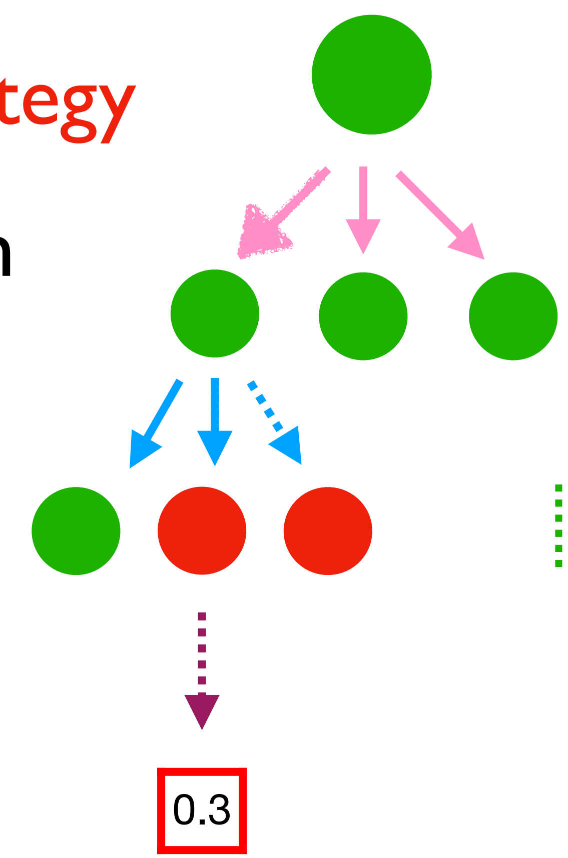


- δ is the width of the confidence interval
- The width is calculated to ensure that $\mu \leq \bar{\mu} + \delta$ and the width decreases when the #pulls get large.

Optimal convergence rate is guaranteed.

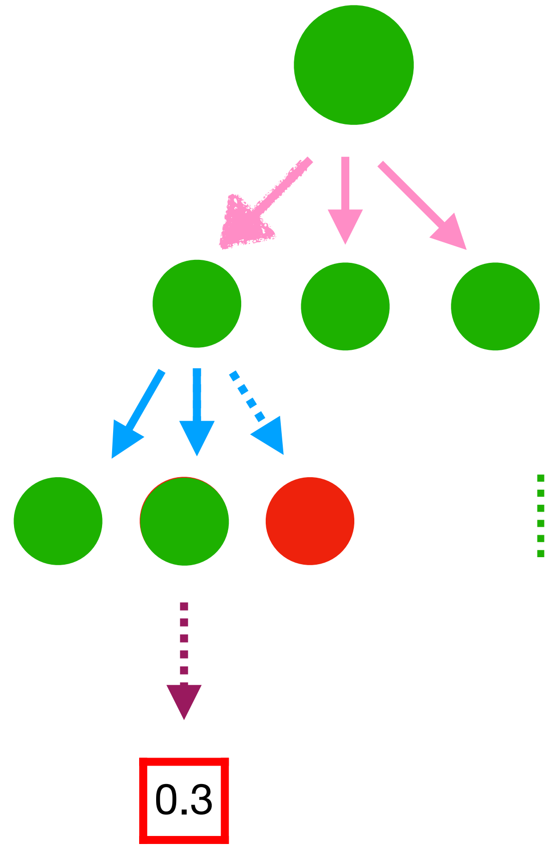
Monte-Carlo Tree Search (MCTS)

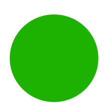
- Nodes of two kinds: visited node, unvisited node.
- During MCTS, we use two strategies: tree and default
- Algorithm: repeat until time or space limit:
  - Selection: choose one node among using tree strategy
  - Expansion: if the node has unvisited child, expand one and put into
  - Simulation: simulate down using default strategy
  - Update: update MC estimation through path
- Output the best action to play



The default strategy is usually random play
The tree strategy is essential:
Deciding the order of search

Bandit Tree Strategy for MCTS



- Node selection: treat  as bandit arms! We can use UCB or others.

Node selection in MCTS is a best arm identification problem in bandit learning, which focuses on exploration in stead of exploration vs. exploitation (why?). Still an active research problem.

Outline: Decision Making (II)

- Adversarial game
  - Two-player zero-sum game
- Deterministic search
  - Minimax search
  - Alpha-beta pruning
- Simulation-based search
  - Monte-Carlo tree search
- Stochastic environment: a prelude
- Take-Home Messages

Take-Home Messages

- Adversarial search
  - Game among multiple rational players
- Two-player zero-sum game
  - Nash equilibrium in two-step games
  - Minimax search and alpha-beta pruning: deterministic search without benefiting from the structure of the search tree
- Search by simulation
  - Monte-Carlo tree search: benefit from exploration of tree structure and random simulation.

Next lecture: deal with unknown stochastic environment with reinforcement learning

Thanks for your attention!
Discussions?

Acknowledgement: Many materials in this lecture are taken from
http://ai.berkeley.edu/lecture_slides.html
https://cs.nju.edu.cn/zlj/Course/Theory_17.html
https://www.davidsilver.uk/teaching/